НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.415.2

«До захисту допущено»
Завідувач кафедри
_____ І.А. Дичка
«___»_____2018 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Модифікований програмний метод безперервної передачі даних в умовах швидкого розгортання мережі для пристроїв з ОС Android»

Виконав:
студент VI курсу, групи КП-71мп
Петрусь Владислав Ігорович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н.,
Олещенко Любов Михайлівна _____

Рецензент: доцент кафедри автоматики та управління
в технічних системах ФІОТ, к.т.н., доцент
Полторак В.П. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2018 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення» («Програмне забезпечення комп'ютерних та інформаційно-пошукових систем»)

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____ 2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Петрусю Владиславу Ігоровичу

1. Тема дисертації «Модифікований програмний метод безперервної передачі даних в умовах швидкого розгортання мережі для пристроїв з ОС Android», науковий керівник дисертації доцент кафедри ПЗКС, к.т.н., Олещенко Любов Михайлівна затверджені наказом по університету від «15» листопада 2018 р. № 4173-с

2. Термін подання студентом дисертації «14» грудня 2018 р.

3. Об'єкт дослідження: процес автоматизованого розгортання та підтримання однорангових бездротових мереж.

4. Предмет дослідження: методи оптимізації маршрутизації та безперервної передачі даних в однорангових бездротових мережах.

5. Перелік завдань, які потрібно розробити:
   - провести аналіз методів маршрутизації та підтримки роботи однорангових бездротових мереж;
   - дослідити роботу існуючих методів та алгоритмів маршрутизації та підтримки роботи однорангових бездротових мереж;
   - запропонувати та обґрунтувати шляхи модифікації маршрутизації та підтримки роботи однорангових бездротових мереж;
   - розробити метод маршрутизації та підтримки роботи однорангових бездротових мереж;
   - виконати програмну реалізацію розробленого методу;
   - виконати порівняння запропонованого методу із існуючими аналогами та надати оцінку отриманих результатів.

6. Орієнтовний перелік публікацій:
   - XI наукова конференція магістрантів та аспірантів «Прикладна математика та комп'ютинг» (ПМК-2018-2);

7. Дата видачі завдання «15» грудня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.02.2018	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.03.2018	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	16.04.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	14.05.2018	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	17.09.2018	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації; підготовка матеріалів доповіді на конференції ПМК-2018.	02.10.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу;	28.10.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	10.11.2018	

Студент В.І. Петрусь

Науковий керівник дисертації Л.М. Олещенко

РЕФЕРАТ

Актуальність. На сьогоднішній день спостерігається постійне зростання обсягів інформації, що генерується людством та кількість розробок у сфері рухомих мобільних пристроїв, зокрема — квадрокоптерів, мобільних обчислювальних машин з бездротовим мережевим інтерфейсом та, відповідно, інформації, яку необхідно передавати між цими пристроями і яка генерується в реальному часі та потребує обробки. Організація таких пристроїв в мережі та підтримка безперервної передачі даних є важливою задачею для багатьох сфер науки та виробництва. Дана задача є актуальною через різноманітність її застосувань, у яких необхідно враховувати зміну вузлів у просторі відносно їх мобільних характеристик, таких як зменшення втрат пакетів під час обміну інформації між вузлами, зменшення часу затримки перерахунку таблиць маршрутизації мережі, покращення синхронізації рухомих об’єктів об’єднаних в мережу.

На даний час існує багато методів безперервної передачі даних в умовах швидкого розгортання бездротових мереж, що відрізняються навантаженням на вузол, кількістю підрахунків та швидкістю виконання. Аналіз існуючих методів визначення нечітких дублікатів показав, що більшість з них характеризується слабкою підтримкою ситуацій, в яких вузли можуть змінювати своє відношення один відносно одного, зокрема, заміни доступності та якості сигналу. Таким чином, розроблення методу безперервної передачі даних в умовах швидкого розгортання бездротових мереж, який би мав покращену підтримку рухомих вузлів однорангової мережі, є актуальною задачею.

Об’єктом дослідження в даній роботі є процес автоматизованого програмного розгортання та підтримання однорангової бездротової мережі на основі пристроїв з ОС Android.

Предметом дослідження є методи програмної оптимізації маршрутизації та безперервної передачі даних в однорангових бездротових мережах.

Метою дослідження є зменшення затримок у вузлах під час маршрутизації пакетів у вузлах, покращення безперервної передачі даних відносно існуючих методів.

Методи дослідження. В роботі використовуються методи комп'ютерного моделювання, статистичні та емпіричні методи.

Наукова новизна роботи полягає в наступному.

Запропонована модифікація програмного методу безперервної передачі даних, що відрізняється від існуючих методів врахуванням зміни відношення вузлів в мережі додаванням мультиплексування передачі даних між вузлами, що дозволяє збільшити точність при побудові таблиць маршрутизації при передачі даних.

Практична цінність отриманих в роботі результатів полягає в тому, що запропонований метод дає змогу виправити недолік існуючих методів безперервної передачі даних в бездротових мережах, що пов'язаний із зміною положення вузлів мережі відносно один одного у мережі. Усунення цього недоліку надає можливість оптимізувати роботу на розгортання бездротових мереж у системах FANET, VANET, MANET та зменшити кількість втрати пакетів під час передачі даних.

На основі розробленого методу розроблено додаток для операційної системи Android для обміну інформацією між мобільними пристроями.

Апробація роботи. Основні положення і результати роботи доповідалися та обговорювалися на X науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, п'яти розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, наведено відомості про апробацію результатів.

У першому розділі сформульовано мету і задачі дослідження; розглянуто сучасні методи безперервної передачі даних в бездротових

мережах; наведено їх переваги та недоліки; розглянуто наявні комерційні програмні продукти, що підтримують безперервну передачу даних в бездротових мережах; розглянуто структуру обміну даними та архітектуру мереж.

У другому розділі обрано базовий метод для модифікації; розглянуто можливості покращення базового методу; запропоновано модифікований метод; визначено алгоритми і структури даних, що необхідні для реалізації запропонованого модифікованого методу.

У третьому розділі описано архітектуру програмного забезпечення, що якнайкраще підходить для реалізації методів безперервної передачі даних; визначено кінцевий алгоритм роботи системи безперервної передачі даних на основі бездротових мереж, що реалізує запропонований метод; описано формат вхідних даних, з якими буде працювати реалізована система.

У четвертому розділі описано методику оцінювання ефективності методів безперервної передачі даних; наведено критерії визначення ефективності запропонованого методу; наведено результати тестування алгоритму, розробленого на основі запропонованого методу; запропоновано шляхи подальшого вдосконалення розробленого методу.

У п’ятому розділі побудовано бізнес-модель, яка обґрунтовує доцільність реалізованого програмного забезпечення та прогнозує його потенційну прибутковість у майбутньому.

У висновках проаналізовано отримані результати роботи.

У додатках наведена копія презентації.

Робота виконана на 92 аркушах, містить 2 додатки та посилання на список використаних літературних джерел з 24 найменувань. У роботі наведено 29 рисунків та 10 таблиць.

Ключові слова: безпровідні мережі, маршрутизація, однорангові мережі, Ad-hoc мережі, Android.

ABSTRACT

Topicality. Today there is a steady increase in the amount of information generated by humanity and the number of developments in the field of mobile mobile devices, in particular – quadrocycles, mobile computers with a wireless network interface and, accordingly, the information that needs to be transmitted between these devices in real time and needs processing The organization of such devices on the network and support for continuous data transmission is an important task for some areas of science and production. This task is relevant due to the diversity of its applications, which should take into account the change of nodes in space due to their mobile characteristics, for example, is to reduce packet loss while exchanging information between nodes and reduce the delay time for recalculation of network routing tables, improve the synchronization of moving objects about connected to the network.

At present, there are many methods of continuous data transmission in the conditions of rapid deployment of wireless networks, differing in load on the node, the number of calculations and the speed of execution. At the same time, an analysis of existing methods for identifying fuzzy duplicates has shown that most of them are characterized by weak support for situations in which nodes can change their attitude towards one another, in particular the replacement of the availability and quality of the signal. Thus, developing a method of continuous data transmission in conditions of rapid deployment of wireless networks, which had improved support for mobile nodes in peer-to-peer networks, is an urgent task.

The object of research in this paper is the process of automated deployment and maintenance of a peer-to-peer wireless network.

The subject of the study is the methods and methods for optimizing routing and continuous data transmission in peer-to-peer wireless networks

The aim of the study is reducing node delays when packet routing in nodes, improving continuous data transmission relative to existing methods..

Research methods. Methods of computer modelling, statistical and empirical methods are used in this work.

The scientific novelty. The proposed modification of the method of continuous data transmission, which differs from existing methods taking into account the change in the ratio of nodes in the network adding multiplexing data transmission between the nodes, which allows you to increase the accuracy of the construction of routing tables in the transmission of data.

The practical value of the results obtained in the work is that the proposed method allows to correct the lack of existing methods of continuous data transmission in wireless networks, which is connected with the change of the position of the network nodes relative to each other in the network. Eliminating this disadvantage provides an opportunity to optimize deployment of wireless networks in FANET, VANET, MANET and reduce packet loss during data transfer.

Based on the developed method, an application for the Android operating system was developed for the exchange of information between mobile devices.

Test work. The main provisions and results of work were reported and discussed at the Xth International Conference of Masters and Postgraduate Students "Applied Mathematics and Computer", PMK-2018.

Structure and scope of work. The master's dissertation consists of an introduction, five sections, conclusions and appendices.

The introduction provides a general description of the work, an assessment of the current state of the problem is made, the relevance of the research direction is substantiated, information about testing the results is given.

In the first section formulated the purpose and objectives of the study; contemporary methods of continuous data transmission in wireless networks are considered; their advantages and disadvantages are presented; Existing commercial software products that support continuous data transmission over wireless networks are considered; the structure of data exchange and network architecture are considered.

The second section chooses the basic method for modification; possibilities of improvement of the basic method are considered; proposed modified method;

the algorithms and data structures necessary for the implementation of the proposed modified method are defined.

The third section describes the software architecture that is best suited for implementing continuous data transfer methods; the definite algorithm of the system of continuous data transmission on the basis of wireless networks, implementing the proposed method; describes the format of the input data, which will be implemented by the implemented system.

The fourth section describes the theoretical basis for evaluating the effectiveness of continuous data transmission methods; the criteria for determining the effectiveness of the proposed method are given; The results of testing the algorithm developed on the basis of the proposed method are given; ways to further improve the developed method are proposed.

The fifth section presents the construction of a business model that justifies the feasibility of the software and predicts its potential profitability in the future.

The conclusions are analyzed the results of work.

The attachments contain a copy of the presentation.

The work is done on 92 sheets, contains 2 supplements and a link to the list of used literary sources of 24 titles. The paper presents 29 figures and 10 tables.

Keywords: wireless networks, routing, peer-to-peer, ad-hoc networks, Android.

РЕФЕРАТ

Актуальность. На сегодняшний день наблюдается постоянный рост объемов информации, генерируемой человечеством и количество разработок в сфере подвижных мобильных устройств, в частности – квадрокоптеров, мобильных вычислительных машин с беспроводным сетевым интерфейсом и соответственно информации что необходимо передавать между этими устройствами, которая генерируется в реальном времени, и требует обработки. Организация таких устройств в сети и поддержка непрерывной передачи данных является важной задачей для некоторых сфер науки и производства. Данная задача является актуальной через разнообразие ее применений, в которых необходимо учитывать изменение узлов в пространстве из-за их мобильную характеристику, например – это уменьшение потерь пакетов при обмене информации между узлами и уменьшение времени задержки пересчете таблиц маршрутизации сети, улучшение синхронизации движущихся объектов о соединенных в сеть.

В настоящее время существует много методов непрерывной передачи данных в условиях быстрого развертывания беспроводных сетей, отличающихся нагрузкой на узел, количеством подсчетов и скоростью исполнения. В то же время, анализ существующих методов определения нечетких дубликатов показал, что большинство из них характеризуется слабой поддержкой ситуаций в которых узлы могут менять свое отношение друг относительно друга, в частности замены доступности и качества сигнала. Таким образом, разработка метода метод непрерывной передачи данных в условиях быстрого развертывания беспроводных сетей, который имел улучшенную поддержку подвижных узлов одноранговой сети, является актуальной задачей.

Объектом исследования в данной работе является процесс автоматизированного программного развертывания и поддержания одноранговой беспроводной сети на основе Android устройств.

Предметом исследования являются програмные методы оптимизации маршрутизации и непрерывной передачи данных в одноранговых беспроводных сетях.

Целью исследования является уменьшение задержек в узлах при маршрутизации пакетов в узлах, улучшение непрерывной передачи данных относительно существующих методов.

Методы исследования. В работе используются методы компьютерного моделирования, статистические и эмпирические методы.

Научная новизна работы заключается в следующем.

Предложенная модификация програмного метода непрерывной передачи данных, отличается от существующих методов учетом изменения отношения узлов в сети добавления мультиплексирования передачи данных между узлами, что позволяет увеличить точность при построении таблиц маршрутизации при передаче данных.

Практическая ценность полученных в работе результатов заключается в том, что предложенный метод позволяет исправить недостаток существующих методов непрерывной передачи данных в беспроводных сетях, связанный с изменением положения узлов сети относительно друг друга в сети. Устранение этого недостатка позволяет оптимизировать работу на развертывание беспроводных сетей в системах FANET, VANET, MANET и уменьшить количество потери пакетов при передаче данных.

На основе разработанного метода разработаны приложение для операционной системы Android, для обмена информацией между мобильными устройствами.

Апробация работы. Основные положения и результаты работы докладывались и обсуждались на X научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2018.

Структура и объем работы. Магистерская диссертация состоит из введения, пяти глав, заключения и приложений.

Во введении дана общая характеристика работы, выполнена оценка современного состояния проблемы, обоснована актуальность направления исследований, приведены сведения об апробации результатов.

В первой главе сформулированы цель и задачи исследования; рассмотрены современные методы непрерывной передачи данных в беспроводных сетях; приведены их преимущества и недостатки; рассмотрены имеющиеся коммерческие программные продукты, поддерживающие непрерывную передачу данных в беспроводных сетях; рассмотрена структура обмена данными и архитектуру сетей.

Во втором разделе избран базовый метод для модификации; рассмотрены возможности улучшения базового метода; предложен модифицированный метод; определены методы и структуры данных, необходимых для реализации предложенного модифицированного метода.

В третьем разделе описано архитектуру программного обеспечения, как нельзя лучше подходят для реализации методов непрерывной передачи данных определены конечный алгоритм работы системы непрерывной передачи данных на основе беспроводных сетей, реализует предложенный метод, описан формат входных данных, с которыми будет работать реализована система.

В четвертом разделе описано теоретические основы оценки эффективности методов непрерывной передачи данных приведены критерии определения эффективности предложенного метода; приведены результаты тестирования алгоритма, разработанного на основе предложенного метода; предложены пути дальнейшего совершенствования разработанного метода.

В пятом разделе приведена построение бизнес-модели, обосновывающая целесообразность реализованного программного обеспечения прогнозирует его потенциальную прибыльность в будущем.

В выводах проанализированы полученные результаты работы.

В приложениях приведена копия презентации.

Работа выполнена на 92 листах, содержит 2 приложения и ссылки на список использованных литературных источников из 24 наименований. В работе приведены 29 рисунков и 10 таблиц.

Ключевые слова: беспроводные сети, маршрутизация, одноранговые сети, Ad-hoc сети, Android.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

OSI (Open Systems Interconnection) – абстрактна мережева модель для комунікацій і розробки мережевих протоколів.

WLAN (*Wireless Local Area Network*) – бездротова локальна мережа.

WNIC (Wireless Network Interface Controller) – мережевий інтерфейс, який підключається до бездротової мережі.

AP (Acces Piont) – бездротові маршрутизатори, є базовими станціями для мережі, які передають та приймають радіочастоти для бездротових пристроїв для зв'яку з ними.

ANET (Ad Hoc Networks) – децентралізована бездротова мережа, яка не має постійної структури.

CBR (Constant Bit Rate) – тип генератору трафіку, що створює потік з фіксованим темпом видачі пакетів, залежить від параметру конфігурації інтервалу.

ВСТУП

Задача побудови мереж з безперервним режимом передачі даних є надзвичайно важливою як у сфері бездротових мереж, так і в загальній інформаційно-обмінній сфері, особливо для рухомих вузлів і зводиться не лише до тривіальної інженерної задачі, що має на даний час багато рішень, але й покриває достатньо великий обсяг проблем та має наукову новизну. Особливо це стосується роботи з рухомими вузлами та побудови на основі такого типу вузлів бездротових мереж та підтримання стану втрати мінімальної кількості пакетів під час обміну інформацією між двома і більше активними агентами мережі.

Для побудови такого типу мереж у деяких випадках використовують мобільні комп’ютери, обладнані бездротовими інтерфейсами, а саме об'єднання декількох мобільних пристроїв з бездротовими мережевими адаптерами в одну мережу передачі даних.

Такого типу мобільні пристрої можуть належати різним особам, тобто єдине централізоване адміністративне управління відсутнє. Крім цього, вузли таких мереж повинні коректно взаємодіяти один з одним на рівні організації як одне ціле. Такими характеристиками володіють мережі для швидкого розгортання, а саме самоорганізовані мережі. Також такі мережі швидко адаптуються до умов змінної характеристики пропускної спроможності мережі.

Однією з фундаментальних проблем управління даними мережами є їх динаміка, та те, що мобільні пристрої обмежені у кількості та в своїх обчислювальних можливостях. Обмеження апаратної частини є достатньо впливовим важелем, який намагаються обійти, застосовуючи різні засоби, але в загальному випадку рішення вимагає залучення більш нових та більш оптимізованих алгоритмів та протоколів, призначених для роботи з мережею. Також існує проблема маршрутизації трафіку всередині мереж та внутрішньої ієрархії передавачів, залучених до системи. Сама система, у свою чергу, зі зростанням кількості вузлів може мати проблему

маршрутизації трафіку системи та балансування навантаженості мережі. Також ці мережі мають проблеми оптимізації пропускної спроможності та управління потужністю. Крім того, їх багатозадачна природа і відсутність фіксованої інфраструктури вводить нові наукові проблеми, такі, як мережева конфігурація, пошук пристроїв і підтримка топології, а також спеціальна адресація і саморуйнування мережі.

Маючи на увазі все вище зазначене, очевидно, що дана проблематика є достатньо перспективною та багато цільовою в рамках мереж, що на фоні нових апаратних можливостей може дати достатньо цікаві результати в імплементуванні розроблених рішень у цій тематиці.

РОЗДІЛ 1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд існуючих методів визначення нечітких дублікатів

1.1.1 Ad hoc On-Demand Distance Vector

Даний алгоритм є базовим та достатньо поширеним серед мереж, що використовують бездротові інтерфейси обміну даними, а саме мереж, що базуються на бездротовому обміну даними як однорангової, так і клієнт-серверної архітектури. Даний алгоритм призначений для динамічної, легко змінюваної, маршрутизації агентів мережі для мобільних ad-hoc безпроводових самоорганізаційних мереж, призначений для MANET мереж. MANET, в свою чергу, є мобільними самоорганізовуваними децентралізованими мережами, що складаються з мобільних вузлів. Причому підтримка як unicast, однонаправлена передача даних одному адресату або маршрутизація пакетів у мережі даних, що не підтримує ширококанальне розповсюдження повідомлень multicast пакетів, є частинним випадком групової передачі даних, при якому створюється мультикаст-груп,а що є отримувачами даного повідомлення. Такий обмін підтримується як на канальному, так і на прикладному рівнях.

Даний протокол є реактивним, що дозволяє реалізовувати доставку повідомлень з генерацією маршруту за необхідністю, причому достатньо важливо мати на увазі, що на відміну від класичних алгоритмів маршрутизації в мережі Інтернет є попереджаючим, тому що не знаходить шляхи маршрутизації трафіку в мережі не залежно від використання маршрутів.

Реалізація даного алгоритму має декілька характеристик:

- розподіленість;
- ітераційність;
- асинхронність.

У рамках даного алгоритму можна зазначити, що кожен вузол веде внутрішню таблицю маршрутизації з інформаційними даними для кожного маршрутизатора або агента мережі.

У AODV за допомогою застосування порядкових номерів при оновленнях маршруту виключена можливість виникнення проблеми «рахунки до нескінченності», притаманна іншим протоколам, які використовують цей алгоритм маршрутизації.

Кожен вузол опитує кожен наступний вузол та просить запитати кожен наступний вузол про те, чи не є поряд від них шуканий вузол.

Протокол обміну даними базується на RREQ пакетах повідомлень, що складаються з наступних полів:

- тип (8 біт) – «1»;
- прапор «J» встановлюється, якщо RREQ передається широкомовно;
- прапор «R» встановлюється при відновленні маршруту;
- прапор «U» встановлюється в тому випадку, якщо порядковий номер вузла-одержувача невідомий;
- 11 біт зарезервовано;
- порядковий номер вузла-одержувача (Destination Sequence Number);
- IP-адреса вузла-ініціатора (Originator IP Address);
- порядковий номер вузла-ініціатора (Originator Sequence Number);
- прапор «G» встановлюється в тому випадку, якщо необхідно встановити не тільки маршрут до вузла-одержувача, а й зворотний маршрут від вузла-одержувача до вузла-ініціатора;
- Hop Count (Кількість переходів) – 8 біт;
- RREQ ID (ідентифікаційний номер повідомлення RREQ);
- IP-адреса вузла-одержувача (Destination IP Address);
- прапор «D» встановлюється, якщо необхідно доставити повідомлення RREQ безпосередньо до вузла-одержувача.

Недоліками такого алгоритму є те, що під час роботи в алгоритмі не передбачено можливість оптимізації на рівні FANET, при обробці ситуації різкої зміни положення тіла в просторі при зміні фізичних характеристик (положення в просторі відносно інших мереж), а також появу перешкод на шляху сигналу.

1.1.2 Open Shortest Path First

Основа ідеї алгоритму базується на можливостях динамічних мереж, а саме на відслідковуванні стану каналу, так званому link-state технологічному процесі, що інкапсулює в собі розповсюджене рішення знаходження найкоротшого шляху, а саме рішення, що базується на алгоритмі Дейкстри, нідерландського науковця у галузі комп'ютерних наук, від однієї вершини до всії інших вершин.

Варто зазначити, алгоритм що володіє наступними характеристиками (рис 1.1):

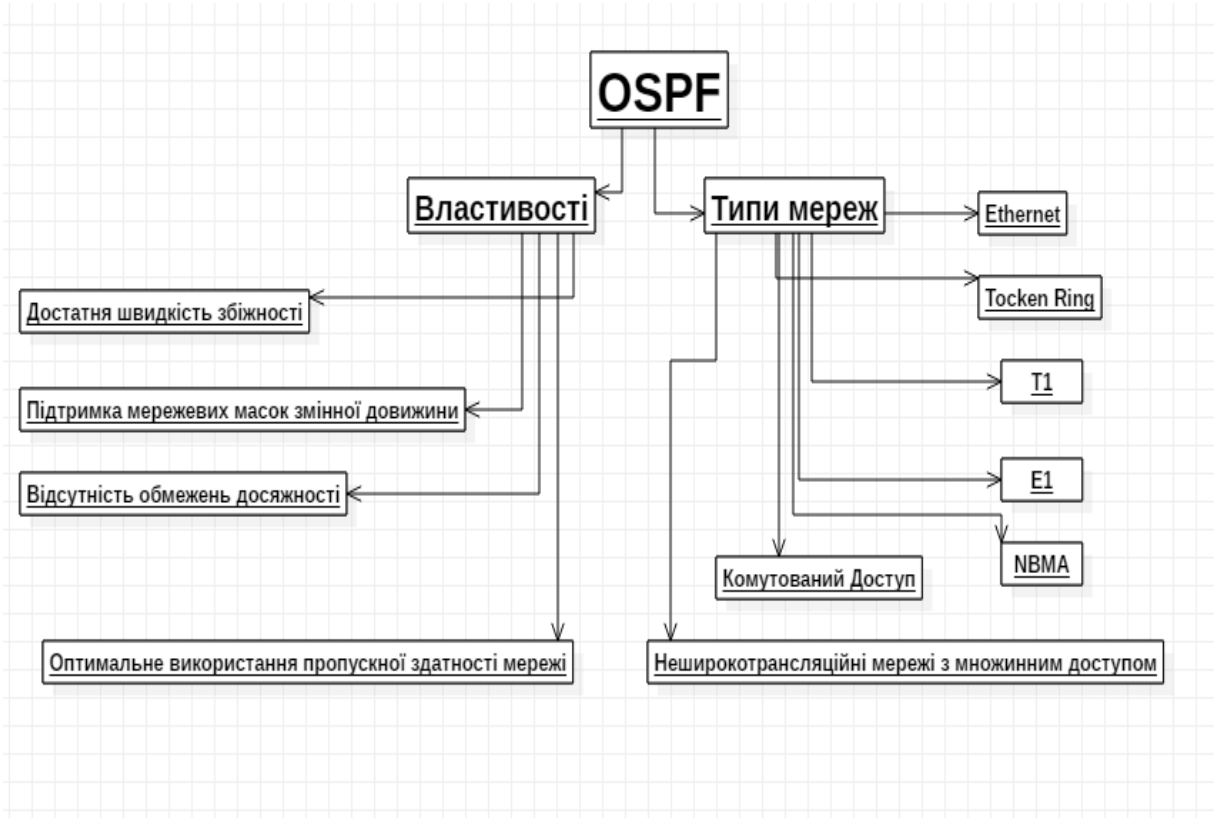


Рис. 1.1 Загальна інформація характеристик OSPF

Являє собою алгоритмічну реалізацію внутрішнього шлюзу Interior Gateway State IGS. Основним підходом є поширення інформації між вузлами про маршрути між вузлами однієї динамічної ad-hoc мережі за допомогою допоміжних проміжних обмінів повідомленнями.

На відміну від попереднього алгоритму, можливі наступні підтримувані режими функціонування мережі:

- широкомовні мережі з множинним доступом (Ethernet, Token Ring);
- точка-точка (T1, E1, комутований доступ);
- нешироковідсильні мережі з множинним доступом (NBMA) (Frame relay).

Сам алгоритм можна описати наступними кроками.

- Після ініціювання мережі вузли починають в асинхронному режимі обмінюватись привітальними пакетами, щоб дізнатись місце знаходження кожного вузла та подудувати загальну карту мережі в рамках зони видимості свого вузла відносно загального графа. Вузли, що дають ресурси на використання загального каналу передачі даних (що в свою чергу можуть бути завантаженими іншими вузлами), створюють ребро взаємодії в рамках графу при заданих конфігураціях.
- Конфігурації такого стану обираються в околі конфігурацій за замовчуванням.
- Встановлення стану сумісності з вершинами поряд. Цей перехід прямо пропорційно залежить від типу маршрутизатора, що починають обмінюватись привітальними пакетами та типом мережі. Вузли, що знаходяться в спряженому стані, починають синхронізацію.

- Починається групове оповіщення вузлів мережі про стан каналу маршрутизаторам, з явними він знаходиться в заданій синхронізації.
- Вузли рекурсивно отримують оповіщення від сусідніх вузлів про стан каналу.
- Після синхронізаційного етапу кожних агент мережі матиме ідентичну копію таблиці маршрутизації в середині кожного вузла.

Перевагою цього процесу є те, що після фінальних синхронізацій і приєднання всіх вузлів до мережі кожен агент починає працювати швидше, використовуючи алгоритм найкоротшого шляху до цільового вузла першим. Варто зазначити що таке застосування використовують тільки для графу без петель, що описує найкоротший шлях до всіх інших відомих агентів, та обирає з її множини відповідний адресат.

Недоліком є те, що ініціалізаційний етап проходить відносно довго і є дуже дорого операцією як за обсягом використовуваної пам’яті, так і за кількістю операцій.

Синхронізаційні таблиці займають достатньо багато пам'яті, і в залежності від величини графу можуть надзвичайно розростатися.

Будь-які зміни в рамках додавання або віднімання вузла потребують перебудови загальної карти мережі та ініціалізації всієї процедури заново.

Також алгоритм не має в собі обробки ситуацій зміни фізичного положення вузла і більш націлений на фіксовані бездротові мережі.

1.1.3 RIP

Даний протокол має достатньо просту алгоритмічну реалізацію, що робить його достатньо привабливим для рішень простих задач. Під простими задачами можна мати на увазі імплементацію в малих комп'ютерних мережах. Як і інші аналоги, реалізує набір функціональності для комп'ютерних мереж для динамічного оновлення маршрутної інформації. Джерелом даних для кожного агента мережі виступають сусідні агенти, що передають за вимогою певну загальну інформаційну характеристику мережі самому вузлу, причому особливістю мережі є направлення передачі інформації та дальність в «стрибках».

Зазвичай такого роду характеристика необхідна для позначення відстані між вузлами. Даний алгоритм не враховує перешкоди, що можуть існувати під час передачі даних в рамках однієї транзакції файлу або повідомлення, що в свою чергу створює ряд проблем для реалізації у рамках швидкого розгортання мережі.

Таблиця 1.1

Таблиця коефіцієнтів складності шляху

Змінна	Значення
n	Кількість вузлів згенерованого шляху
S	Довжина ребра (фізична характеристика)
N	Загальна кількість вузлів у графі
b	Стійкість сигналу мережі
$\mathcal{J}$	Алгоритмічна потужність вузла, необхідна на обробку пакету даних та передачу його далі
P	Складність перечаді даних всередині мережі

У рамках цього алгоритму необхідно ввести позначення ребра з певною фізично-інформаційною характеристикою між двома агентами графу мережі, за якою відбувається передача пакетів.

Основна логіка підтримки безперервної передачі даних полягає в мінімізації транзакційних моментів передачі даних, врахування складності шляху, який прямо пропорційний до дальність шляху (1.1) на основі коефіцієнтів, наведених в таблиці 1.1.

.

$$P = n \times \frac{\sum_{1}^{n} S_i}{N} \times b \times \vartheta \qquad (1.1)$$

Варто зазначити, що даний алгоритм має один недолік в рамках своєї простоти, а саме максимальна кількість стрибків у рамках роботи мережі є 15 ребер завдяки внутрішній матричній реалізації. Кожен вузол раз в 30 секунд надсилає в загальну мережу пакети з інформацією про карту мережі вузлам. Операція такого роду надзвичайно навантажує мережу, адже потребує передачі великої кількості трафіку. Структуру такого інформаційного пакету можна побачити на рис1.2, та таблиці 1.2

Натомість плюсом в даній ситуації є те, що протокол є простим для конфігурації.

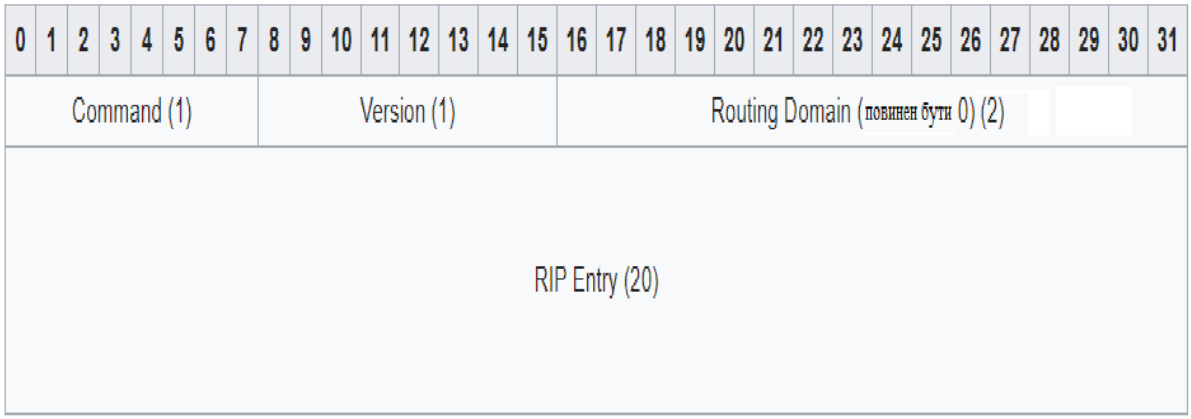


Рис. 1.2. Тіло пакету обміну передачі даних RIP

Таблиця 1.2

Формат PIR пакету

Значення	Опис
Command	Біт, що конфігурує значення блоку інформації, що передається протоколом через мережу зв’язку без встановленого з’єднання попередньо в рамках взаємодії графу мережі. 1 – запит 2 – відповідь
Entity	Запис або пакет, що несе маршрутну інформацію та займає від до 25-ти записів.
Routing Domain	Унікальний ідентицікатор мережевої системи, котрому належить дане повідомлення. Зазвичай використовують згенероване значення, прив’язане до самогоо вузла. Використовується, коли багато каналів під’єднані до одного і того ж каналу. Необхідний для дескретності повідомлень та відділення згеренованих повідомлень у рамках каналу передавача, або в рамках каналу-перехолювача повідомлення.
Version	Номер версії, або формат пакету.

Така специфікація алгоритму дозволяє з достатньо високою швидкістю передавати інформацію та підтримувати постійне бездротове з'єднання у рамках однієї сесії.

Оскільки синхронізаційні пакети повідомлень мережі розсилаються усім агентам, що мають підключення до загального графу, повідомлення потрібно розділяти на притаманні своїм вузлам та не притаманним графові, або іншим автономним системам.

1.1.4 EIGRP

Алгоритм розроблений компанією Cisco на основі іншого фундаментального алгоритму для функціонування в межах автономної системи.

Відрізняється тим, що має достатньо широкий спектр можливостей конфігурації мережі та можливості вибору різних компонентів для обрахунків. Компоненти для обрахунку наведені в таблиці 1.3.

Таблиця 1.3

Таблиця компонентів джерел для обрахунку маршуртизаційних таблиць мережі

Назва	Опис
Bandwidth	Мінімальна пропускна здатність для даного маршруту (а не сума цін (cost) на відміну від OSPF).
Delay	Сумарна затримка на всьому шляху маршруту.
Reliability	Найгірший показник надійності на всьому шляху маршруту, заснований на keepalive.
MTU	Найгірший показник завантаженості інтерфейсу на всьому шляху маршруту, заснований на кількості трафіку, що проходить через інтерфейс і налаштованому на ньому параметрі bandwidth.
Loading	Мінімальний розмір MTU на всьому шляху маршруту.

Алгоритм можна побачити на зазначеному нижче рисунку (рис 1.3)

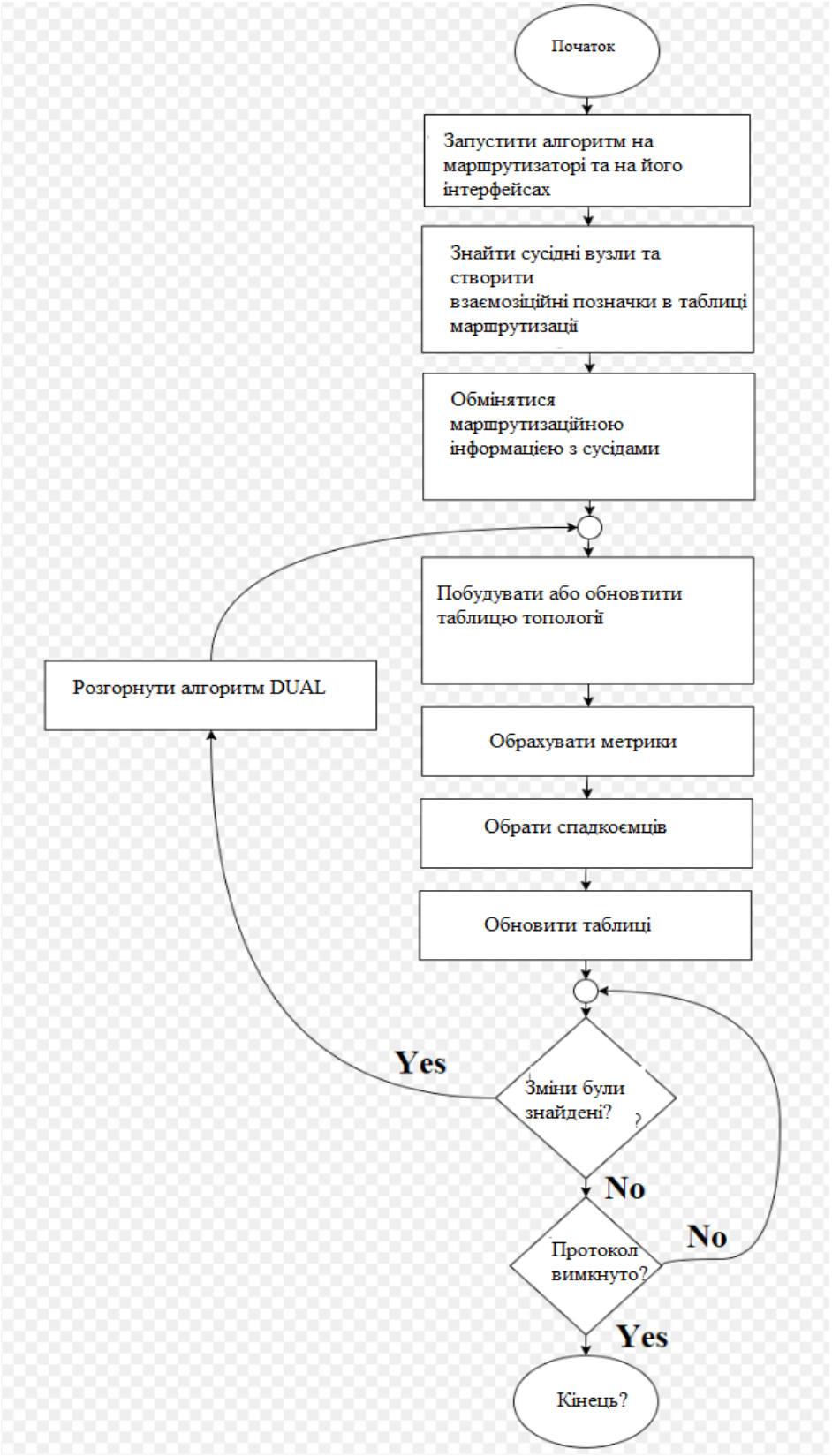


Рис 1.3. Алгоритм EIGRP

Варто зауважити, що ускладнення формули обчислення метрик призводить і до ускладнення розуміння метрики адміністратором. Хоча багато прихильників OSPF вважають, що, за інших рівних, EIGRP відпрацьовує зміну топології повільніше ніж OSPF, але це не так, оскільки при малій кількості маршрутизаторів EIGRP відпрацьовує швидше, а при ускладненні схеми архітектура протоколу OSPF вимагає більш ретельного впровадження (створення зон і між зонних відносин), що також уповільнює обмін маршрутами і ускладнює кількість обчислень, необхідних для вибору кращого маршруту. В результаті EIGRP працює порівняно однаково, а в деяких найпростіших або навпаки більш складних топологія навіть швидше ніж інші, існуючі на даний момент, протоколи маршрутизації.

Критичним недоліком такого алгоритму є те, що він надзвичайно обмежений у використанні апаратури CISCO і не сумісний із жодним із аналогів існуючих на даний час. Алгоритм є обмеженим з точки зору сумісності та вимагає достатньо великих витрат під час імплементації в інфраструктуру користувача. Також така апаратура не має практик використання в рухомих пристроях, через свою експериментальність та обмеженість в ресурсах. Варто зазначити, що даний алгоритм не був протестований значною мірою на швидко рухомих вузлах.

Відмінною доробкою в рамках цього буде якраз імплементація покращення для швидко рухомих вузлів.

1.2 Огляд існуючих комерційних програмних продуктів

На даний момент велика кількість комерційних мережевих продуктів використовують в своїх рішеннях алгоритмічну базу маршрутизації. Підтримка стабільного сигналу для клієнтів є найголовнішою задачею наступних видів бізнесу:

- телекомунікації;
- медіа;
- менеджери;
- хмарні обчислення;
- зв’язок;
- туризм;
- охорона здоров’я та безпека;
- моделювання.

Ці та багато інших видів послуг об’єднує необхідність постачати клієнтові постійне та якісне з’єднання до мережі, необхідність синхронізувати дані та зберігати стани не тільки в одному місці, та й в багатьох місцях одночасно.

Програмні продукти так чи інакше завжди використовують певну програмну реалізацію мережевого балансування та маршрутизацію, адже це є основою роботи будь-якої мережі. Далі неведено декілька прикладів програм.

1.2.1 Utorrent

Інтернет-сервіс " Utorrent " (рис 1.4) пропонує набір послуг, що в сукупності реалізують технологію передачі даних всередині мережі на основі розподілених місць зберігання інформації та сумісної валідації та покращення швидкодії кожного вузла як окремої машини з можливостями завантаження файлів в мережу та вивантаження на конкретну машину деякого контенту за потребою користувача. Варто зазначати, що повинна

існувати хоча б одна копія файлу або групи файлів, що необхідно розділити всередині об’єднаної групи користувачів, що зібралися, для того, щоб отримувати та зберегти дану інформацію.

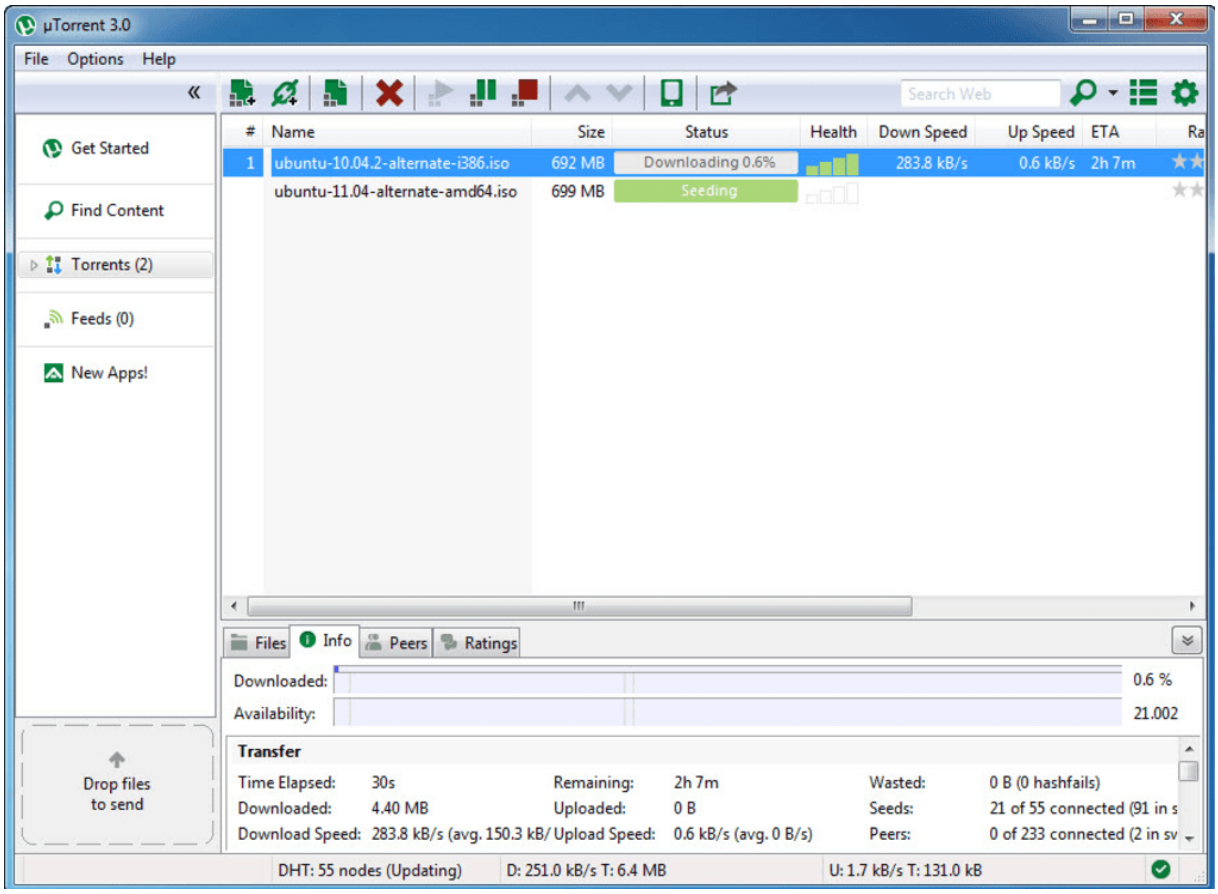


Рис. 1.4 Система utorrent

Функціональне ядро торента використовує унікальні алгоритми, розроблені вченими та інженерами всього світу, що забезпечує:

- швидкий і ефективний пошук;
- завантаження файлу;
- створення файлів-ключів на певний файл;
- групове завантаження файлів;
- можливість використання копії файлів в мережі при зникненні початкового джерела даних за необхідністю;
- підтримку можливості вибору для користувача завантаження тільки тих файлів, які йому необхідні;

- можливість функціонування мережі без існування основного сервера;
- можливість peer-2-peeer режиму роботи.

Стратегічним завданням є підвищення доступу інформації для кожної людини, можливість децентралізованого та швидкого зберігання інформації, можливість балансування мережі за рахунок вузлів, що знаходяться поряд та забезпечити наступні можливості для користувача: швидкий і ефективний пошук, завантаження файлу, створення файлів-ключів на певний файл, групове завантаження файлів, можливість використання копії файлів в мережі при зникненні початкового джерела даних за необхідністю, підтримки можливості вибору для користувача завантаження тільки тих файлів, які йому необхідні, можливість функціонування мережі без існування основного сервера, що в цілому робить даний продукт доволі цікавим для кожного користувача. Але варто зазначити, що торенти дуже прив’язані до мережі Інтернет, і за межами неї дуже рідко себе позиціонують.

1.2.2 Technitium Bit Chat

Technitium Bit Chat – це безпечний одноранговий (p2p) веб-переглядач з відкритим кодом, призначений для забезпечення повного шифрування. Основною метою розробки цього миттєвого обміну повідомленнями є забезпечення конфіденційності, що досягається за допомогою криптографії. Він може використовуватися через Інтернет і приватні мережі мережі для обміну миттєвими повідомленнями та передачі файлів. Архітектура розроблена з урахуванням принципу безпеки. Під час використання Bit Chat відсутні мета-дані. Єдине, що ми знаємо, це електронну адресу користувача, зареєстрованого для цифрового сертифіката.

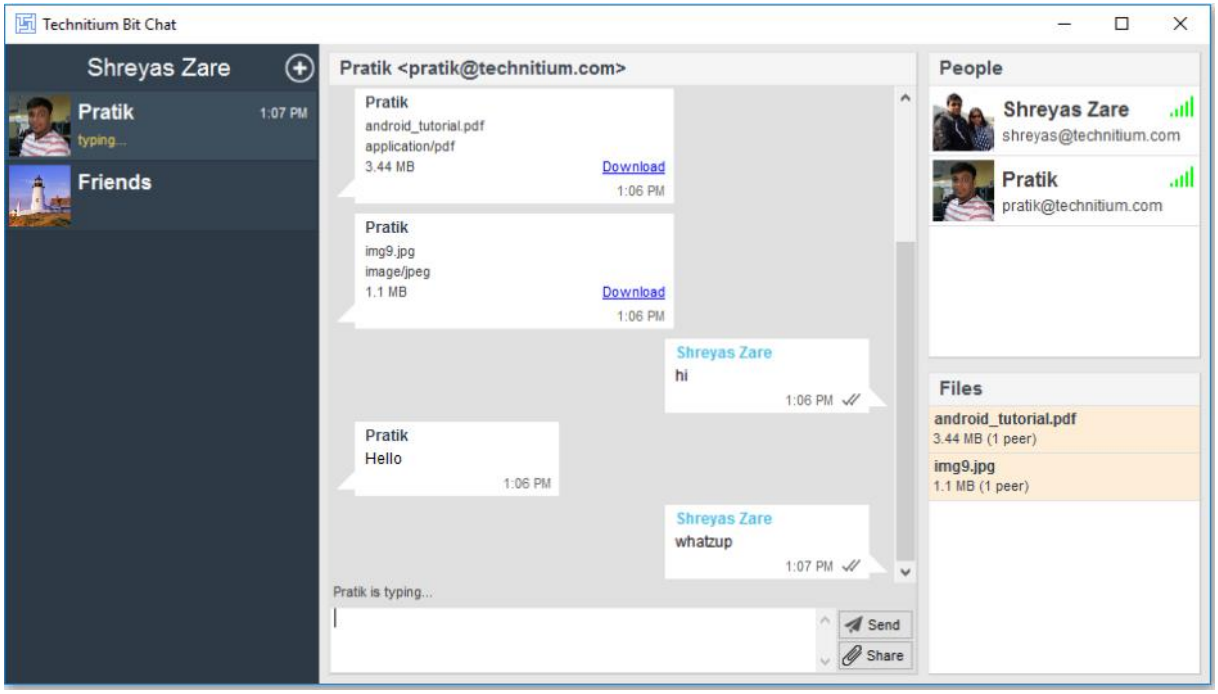


Рис 1.4.1 Меню Technitium Bit Chat

Цифровий сертифікат повідомляє, що електронна адреса перевірена за аналогією до будь-яких сертифікованих SSL сертифікатів, виданих веб-сайтам.

1.2.3 Briar

На даний момент вже перший стабільний варіант вийшов з месенджера Briar. Він доступний для пристроїв Android із Google Play або F-Droid. Цей випуск слідує перевірці безпеки та 10-місячному публічному бета-періоду, протягом якого було виправлено багато помилок та отримано багато відгуків.

Розвиток Briar буде продовжуватися за допомогою Фонду відкритих технологій, який раніше підтримував цей проект як частину своєї місії щодо сприяння свободі Інтернету у всьому світі. Нові можливості, заплановані на 2018 рік, спрямовані на вирішення найбільш нагальних потреб громади. Вони включають можливість додавання контактів віддалено, без попередньої перевірки особистостей. На рисунку 1.5 можна побачити логічну архітектуру додатку.

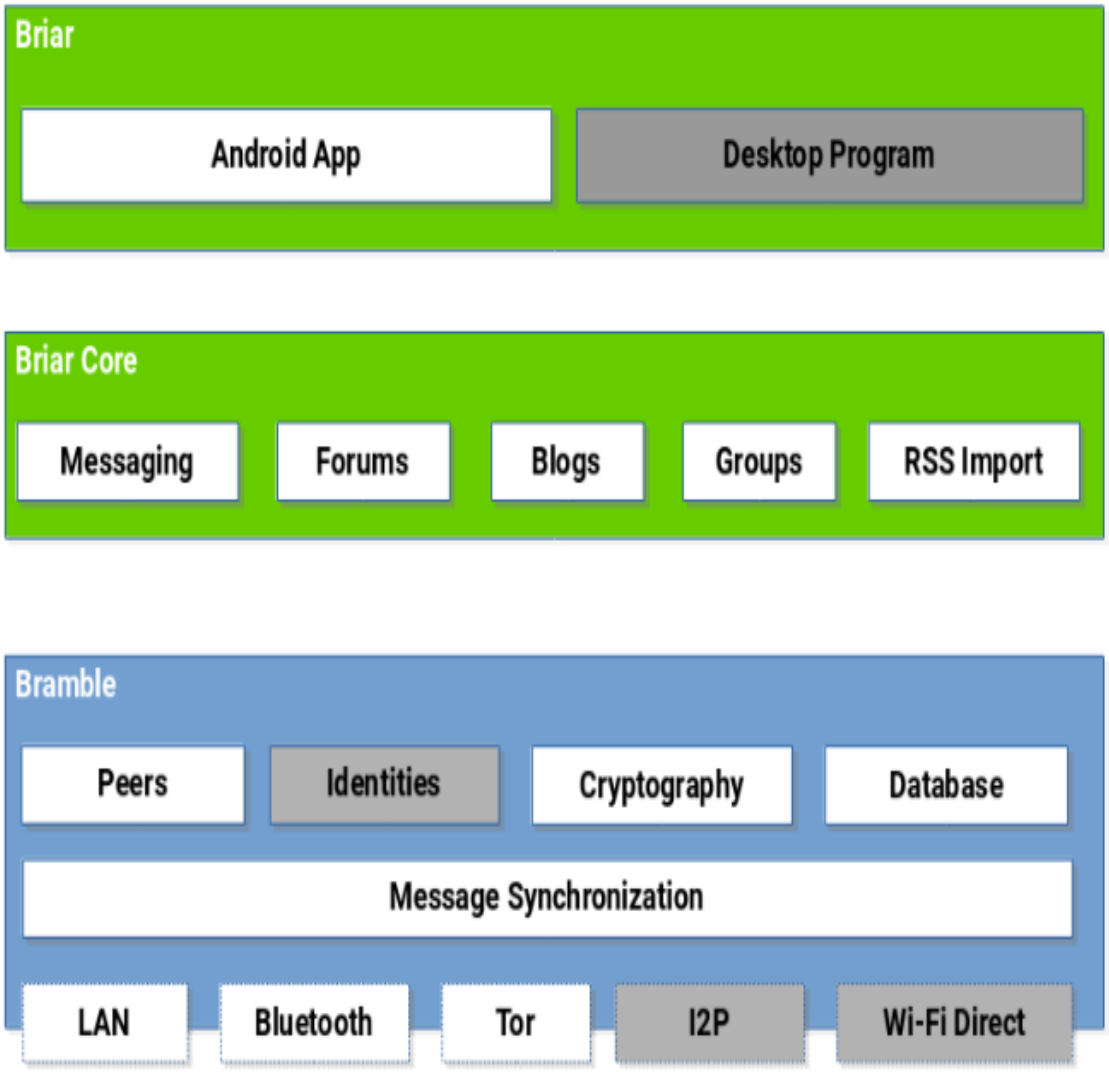


Рис.1.5 Логічна архітектура додатку Briar

Також планується вкладення зображень для повідомлень та поліпшення часу роботи акумулятора. Під час цієї роботи команда Briar також покладе основи для майбутнього настільного клієнта. Багато людей потребують версію Briar для iOS, але приведення P2P Messenger до iPhone є складним, оскільки Apple не дозволяє довгоживучих мережевих з'єднань у фоновому режимі, які необхідні для обміну повідомленнями P2P. Німецький фонд прототипів підтримує Briar у розробці функції, яка дозволить користувачам моментально надсилати один одному повідомлення, навіть якщо вони не є в Інтернеті одночасно. Це підвищить

доступність користувачів до додатку та їх кількість та покращить термін служби акумулятора, а саме зменшить навантаження на споживання електроенергії користувачами.

1.3 Висновок

Задача побудови мереж з безперервним режимом передачі даних являє собою надзвичайно важливу задачу в сфері комп'ютерних мереж як в сфері бездротових мереж, так і в загальній інформаційно-обмінній сфері, особливо для рухомих вузлів і зводиться не лише до тривіальної інженерної задачі, що має в собі вже на даний час безліч рішень зводиться до побудови бездротової мережі та підтримання стану втрати мінімальної кількості пакетів під час обміну інформацією.

Однією с головних проблем є управління даними мережами є їх динаміка, та те, що мобільні пристрої обмежені у кількості та в своїх обчислювальних можливостях. Також існує проблема маршрутизації трафіку всередині мереж та внутрішньої ієрархії передавачів, залучених до системи. Також ці мережі мають проблеми оптимізації пропускної спроможності та управління потужністю. Крім того, їх базатозадачна природа і відсутність фіксованої інфраструктури вводить нові наукові проблеми, такі, як мережева конфігурація, пошук пристроїв і підтримка топології, а також спеціальна адресація і саморуйнування мережі.

Розглянуваши існуючі рішення та існуючі продукти. що базуються на таких рішеннях, можна зазначити та виділити основні напрямки роботи та розвитку вже існуючих алгоритмів та рішень, що потребують певних виправлень та імплементаціях як ін'єкційного типу, так і загально нового підходу.

РОЗДІЛ 2 ФОРМУЛЮВАННЯ МОДИФІКАЦІЇ ПРОГРАМНОГО МЕТОДУ ПІДТРИМКИ МАРШРУТИЗАЦІЇ ТА БЕЗПЕРЕРВНОЇ ПЕРЕДАЧІ ДАНИХ

2.1 Аргументація вибору базового методу безперервної передачі даних

В результаті розгляду теоретичних методів підтримки безперервної передачі даних та маршрутизації в умовах швидкого розгортання виявлено, що їх можна поділити на 3 головні категорії:

- хопові;
- динамічні;
- Bandwidth.

За алгоритмічною основою на:

- адаптивні та неадаптивні;
- глобальні та децентралізовані;
- статичні та динамічні.

За базовими вимогами:

- стабільності;
- оптимальності;
- точності;
- простоти;
- надійності;
- продуктивності.

Дані характеристики показують список важелів, що можуть вплинути на розробку покращень нового алгоритму.

В рамках розробки та в рамках підтримки режиму швидкого розгортання, необхідно базуватися на динамічних алгоритмах. Динамічні алгоритми в свою чергу забезпечують сумісність з динамічними мережами, та зміні стану в залежності від конфігурацій мережі, та від зовнішнього впливу на мережу, за час функціонування графу та взаємодії вузлів мережі.

Також алгоритм повинен мати в собі адаптивні характеристики в комбінації з децентралізованою архітектурою. Дана умова необхідна для конфігурування графу під час додавання та віднімання вузлів або робочих агентів мережі, також для постійного конфігурування мережі у рамках оновлення та зміни характеристик ребер графу. Децентралізована архітектура забезпечує зберігання загальної картини в комбінації з унікальним станом кожного вузлу в самій мережі.

Також за базовими вимогами необхідно обрати алгоритм з найбільшою продуктивність та одночасно з якомога більшим кількістю інших характеристик, що не будуть конфліктувати між собою, наприклад, мати в собі продуктивність, цей стан вимагає достатньої кількості проміжних валідацій, що можуть вимагати додаткового процесорного часу на обробку таких ситуацій, що в кінцевому випадку значно знизить продуктивність самого алгоритму.

Під даний опис підпадають декілька алгоритмів, а саме AODV та DSR (Dynamic Source Routing) алгоритми, що схожі за характеристиками, але різні за своєю природою. Варто зазначити і інші характеристики, що можуть вплинути на вибір базового методу. Дані характеристики наведені на рисунку 2.1.

Важливим моментом у даній схемі є те, що використання неадаптивних алгоритмів у рамках одного вузла не впливають на поточний стан стану загального графу мережі, усі маршрути розраховуються до початку використання мережі та записуються в відповідних таблицях. Будь-яка зміна в рамках мережі, а саме додавання або віднімання вузлу, переконфігурація мережі є повністю неприпустима в рамках даної динамічної мережі, хоча одночасно сама мережа повністю підтримується та може бути в робото здатному стані.

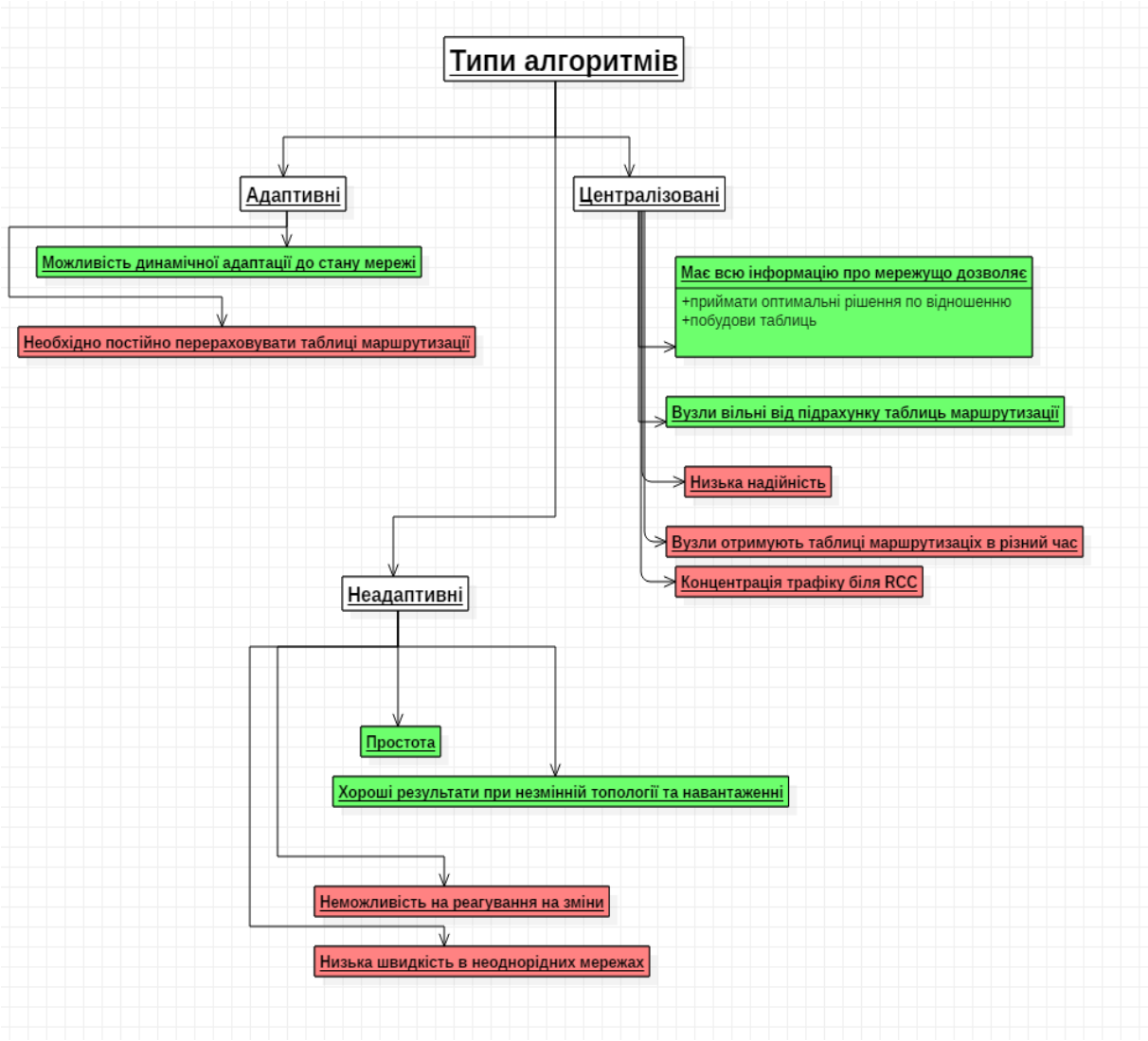


Рис. 2.1. Типізація алгоритмів в залежності від архітектури

Проаналізувавши вище зазначені дані, можна зробити висновки, що базуватися на алгоритмічно – адаптивній основі є необхідним кроком. Причому з розподіленою основою, згідно таблиці 2.1 можна побачити, що розподілена основа дає необхідну гнучкість мережі, відображає та включає обробку змін, що можуть відбутися в ході роботи мережі та побудови проміжних таблиць. Недоліком є складність обрахунків при масштабуванні мережі та під час обрахунків нових вузлів для передачі трафіку через них в рамках покращення пропускної величини, при завантаженні певних вузлів максимальною кількістю трафіку.

Таблиця 2.1

Вибір типу адаптивного алгоритму

Централізований	
Опис	В рамках функіонування мережевого граву існує певна вершина, що отримує інформацію зі всіх вузлів та має найбільшу або достатню обчислювальну вільну потужність для побудови та підрахунків. Зазвичай це довжина черги повідомлень та характеристика завантеженості ліній. Таку точку називають RCC, що відповідає за збір інформації обрахунку оптимальних припустимих та достатніх за швидкістю маршрутів для передачі повідомлення для кожного вузла в мережі та синхронізація вузлів відносно нової заданої інформації за допомогою розсилки.
Переваги та недоліки	Переваги: • RCC володіє всією інформацією і може створювати «ідеальні» маршрути; • вузли звільнені від необхідності розрахунку таблиць маршрутизації. Недоліки: • низька надійність; • раз за декілька ітерацій роботи алгоритму необхідно синхронізовувати таблиці маршрутизації; • можливі колізії при розділенні мережі; • вузли отримують інформацію у різний час; • трафік концентрується біля синхронізаційних точок.

Розподілений алгоритм має ряд переваг, таких як самоорганізація та простота реалізації, проте є проблеми при масштабуванні мережі ,що можуть мати вирішення. Також об’єктом роботи можуть бути проблеми при зникненні бо знищенні одного з вузлів мережі та обрахунок до нескінченності, коли пакет йде по мережі поки не обійде кожний вузол, при цьому можливе зациклювання всередині мережі. При реалізації continuous синхронізації вузлів цей алгоритм отримує надзвичайно великий приріст продуктивності при додаванні вузлів у режимі реального часу. Відповідно так як AODV відповідає всім цім вимогам, можна прогнозувати, що додаючи ряд покращень в сам алгоритм, можна отримати значні покращення продуктивності.

2.2 Можливості покращення обраного методу

Для визначення можливостей для покращення базового методу доцільно розглянути детально базовий метод AODV.

Алгоритм, як зазначено в розділі 2.1, відноситься до дистацнційно-векторних алгоритмів, що робить його легким для розширення.

Алгоритм такого роду є розподіленим, адже обрахунки не зберігаються в конкретному місц,і а прозподілені в кластерах певних груп вершин графу, також передача повідомлень може не бути широко розповсюдженою завдяки асинхронності операцій в цілісній структурі мережі. Архітектурно задано, що кожний вузол має певну сутність, що в свою чергу, може містити інформацію про табличні величини маршрутизаційних даних з однім записом для кожного. Структура даних являє собою вектор структур, що зберігають в собі декілька компонентів, а саме:

- обрана лінія;
- дистанція.

Під час проміжних обчислень вузла виникають задачі оцінки відстані або інших характеристик, таких як:

- дистанції;
- кількості стрибків;
- час затримки;
- довжину повідомлень, що чекають на відправку всередині графу.

В результаті після закінчення ітерацій можна визначити шлях та перебудувати відповідно до формули (2.1) відповідні шляхи, де наступні величини можена виразити за допомогою зміних: X – це сусідній агент мережі, ρ – ймовірний шлях (його довжина), X_i – це припускання щодо довжини шляху до вершити і від самого себе. Якщо агент знає довжину шляху X до і-того вузла мережі, проходження якого який може тривати час m, то вона може бути представлена:

$$\rho = X_i + m \tag{2.1}$$

Схема роботи зображена на рис 2.2.

Апаратна підтримка даного алгоритму реалізована наборами драйверів для мережевої апаратури та забезпечує приріст пропускної величини трафіку каналів і унеможлює врахування трафіку в мережах такого роду через асинхронність. Також відомою проблемою є низька швидкодія стосовно випадків зациклювання всередині вузлу через обмеженість реалізації структур даних збереження маршрутизаційної таблиці.

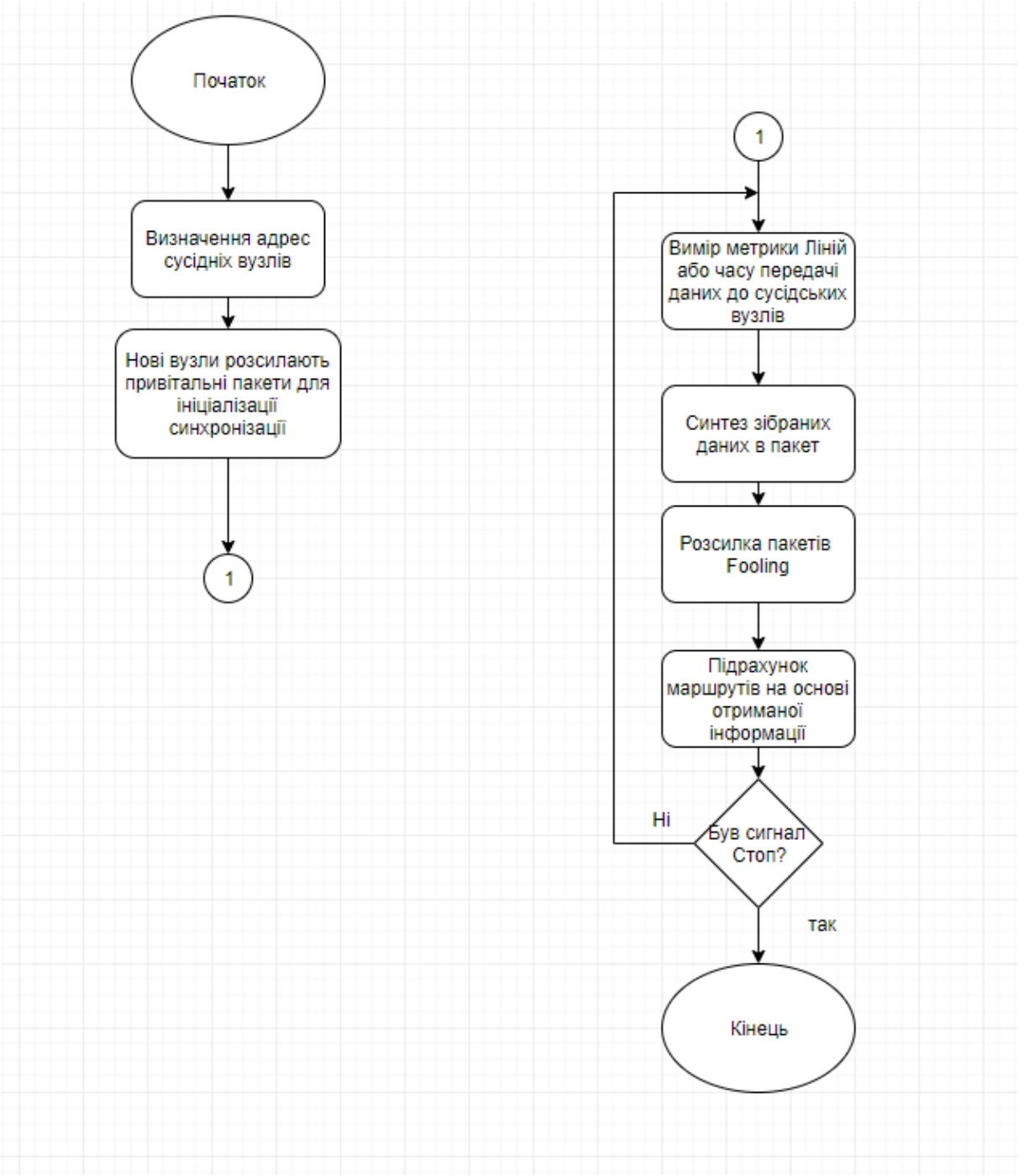


Рис. 2.2. Алгоритм AODV

2.3. Модифікація методу

З огляду на дані попереднього пункту, можна зазначити, що покращення через свою природу необхідно робити в напрямку зміни архітектури протоколу та алгоритму підтримки безперервної передачі даних в однорангових мережах за допомогою комбінації апаратних та алгоритмічних покращень у зв’язку з розвитком апаратних можливостей

мобільних пристроїв з надзвичайно великою швидкістю за останній період часу.

Для покращення даної ситуації запропоновано ряд оптимізацій, де кожному вузлу присвоюється ідентифікатор мережі, що автоматично вираховується під час підключення до неї. Для кожного вузла розраховуються затрати та шлях до сусіднього вузла, ці дані зберігаються в таблиці маршрутизації. Частково кожний вузол може пріорітизувати найдоступніші вузли, які розташовані найближче та мають найменше навантаження та синхронізувати таблиці маршрутизації, виключаючи дублікати.

При кешуванні результату час затримки пакету перед відправкою отримувачеві зменшиться. Цей механізм дозволяє швидко відновлювати маршрути у випадку порушення одного із внутрішніх зв'язків.

При використанні протоколу HTTP необхідно мультиплексувати потоки за допомогою бінарного шару фреймів, що дозволяє передавати пакети в обидві сторони без затримок між потоками. Це дає декілька переваг: паралельні мультиплексовані запити на передачу пакетів даних не блокують один одного; при зберіганні ресурсів на декількох вузлах швидкість зростає.

При стисканні заголовків передачі даних методом HPACK досягається зменшення службового трафіку.

Для «гарячих» вузлів застосовано request-pipe на рівні обробки черги повідомлень. Вузол відслідковує чергу та ігнорує дублюючі повідомлення шляхом попереднього ідентифікування кожного бінарним значенням, що складається з timestamp (4 байти), ідентификатора вузла (3 байти) та лічильника (3 байти). Така модель дозволяє нам створювати до 220 унікальних повідомлень в секунду. Після сегментування пакету та ідентифікатора можна зрівняти пакети та ідентифікатори на дублювання.

2.4 Особливості алгоритму реалізації модифікованого методу

Отже, з огляду на запропоновану модифікацію, можна зробити висновок, що реалізація модифікованого методу підтримки безперервної передачі даних в умовах швидкого розгортання мережі в динамічно змінних системах на основі мобільних пристроїв з ОС Android потребує алгоритмів та структур даних, що забезпечують швидкий пошук на великих об’ємах повідомлень та пакетів. Задача пошуку інформації в програмній інженерії є достатньо складною і актуальною задачею, тому варто зупинитись на описі цієї задачі.

Необхідно додати обробку ситуацій роздробленості мереж не рівномірного розподілення кожної, це можна здійснити способом індексування повідомлення вузла, автоматичного генерування та додавання відслідковування дублювання.

Застосування кешування може прискорити роботу алгоритму.

Необхідна зміна структури даних, так як операція сканування може займати надзвичайно багато часу в рамках пошуку цільового вузла.

Необхідно також використовувати новий підхід що до обробки нових структур згенерованих даних.

2.5 Висновок

В даному розділі було проведено грунтовну роботу в напрямку дослідження розробки методу підтримки безперервної передачі даних в напрямку оптимізації маршрутизації та балансування вузлів мережі для рухомих MANET мереж. Було виконане наступне:

- вибір базового методу;
- дослідження можливого покращення алгоритму;
- опис модифікації методу;
- опис можливих особливостей реалізацій модифікованого методу.

РОЗДІЛ 3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ МЕТОДУ

3.1 Опис засобів розробки програмного забезпечення

З огляду на те, що більшість мобільних телефонів обладнані операційною системою Android ОС та інколи мають поганий Інтернет-зв'язок, було вирішено реалізовувати програмний продукт у вигляді Android додатку, що постачається з усіма необхідними компонентами, необхідними для повноцінної роботи застосунку, а саме:

- можливістю передачі текстових даних в якості чату;
- під'єднанням та роз'єднанням вузлів;
- підтримки балансування в залежності від значних змін конфігурації мережі;
- роботи з багатьма вузлами;
- схеми збереження тимчасових повідомлень;
- можливістю масштабування кількості клієнтів чату;
- режиму безпечного підключення вузлів в мережу.

Пропонується розглянути найпопулярніші інструменти для створення Android застосунків

3.1.1. C#

C# є високорівневою мовою програмування, що забезпечує розробника усім інструментарієм для роботи з операційною системою Windows та можливостями роботи з платформою .Net, що швидко розвивається та підтримує усі новітні можливості роботи сучасних компіляторів та інтерпретаторів.

На сьогоднішній момент мова програмування C # один з найпотужніших, що швидко розвиваються і затребуваних мов в IT-галузі. На даний момент на ньому пишуться найрізноманітніші програми: від невеликих десктопних програмок до великих веб-порталів і веб-сервісів, які обслуговують щодня мільйони користувачів.

У порівнянні з іншими мовами C # досить молодий, але в той же час він вже пройшов великий шлях. Перша версія мови вийшла разом з релізом Microsoft Visual Studio .NET в лютому 2002 року. Поточною версією мови є версія C # 7.0, яка вийшла в 7 березня 2017 року разом з Visual Studio 2017.

C # є мовою з Сі-подібним синтаксисом і близький в цьому відношенні до C ++ і Java. Тому, якщо ви знайомі з одним з цих мов, то опанувати C # буде легше.

Використання традиційних мов .NET-платформи – C # і VB.NET для створення логіки додатка

Можливість декларативного визначення графічного інтерфейсу за допомогою спеціальної мови розмітки XAML, заснованому на xml і представляє альтернативу програмному створення графіки та елементів управління, а також можливість комбінувати XAML і C # / VB.NET

Незалежність від розширення екрану: оскільки в WPF всі елементи вимірюються в незалежних від пристрою одиницях, додатки на WPF легко масштабуються під різні екрани з різною роздільною здатністю.

Нові можливості, яких складно було досягти в WinForms, наприклад, створення тривимірних моделей, прив'язка даних, використання таких елементів, як стилі, шаблони, теми і ін.

Хорошу взаємодію з WinForms, завдяки чому, наприклад, в додатках WPF можна використовувати традиційні елементи управління з WinForms.

Багаті можливості по створенню різних додатків: це і мультимедіа, і двомірна і тривимірна графіка, і багатий набір вбудованих елементів управління, а також можливість самим створювати нові елементи, створення анімацій, прив'язка даних, стилі, шаблони, теми і багато іншого

Апаратне прискорення графіки – незалежно від того, чи працюєте ви з 2D або 3D, графікою або текстом, все компоненти програми транслюються в об'єкти, зрозумілі Direct3D, і потім візуалізуються за допомогою процесора на відеокарті, що підвищує продуктивність, робить графіком більш плавною.

Створення додатків під безліч ОС сімейства Windows – від Windows XP до Windows 10

У той же час WPF має певні обмеження. Незважаючи на підтримку тривимірної візуалізації, для створення додатків з великою кількістю тривимірних зображень, перш за все ігор, краще використовувати інші засоби – DirectX або спеціальні фреймворки, такі як Monogame або Unity.

Також варто враховувати, що в порівнянні з додатками на Windows Forms обсяг програм на WPF і споживання ними пам'яті в процесі роботи в середньому трохи вище. Але це з лишком компенсується більш широкими графічними можливостями і провишенной продуктивністю при відображенні графіки.

Коли говорять C #, нерідко мають на увазі технології платформи .NET (WPF, ASP.NET). І, навпаки, коли говорять .NET, нерідко мають на увазі C #. Однак, хоча ці поняття пов'язані, ототожнювати їх невірно. Мова C # був створений спеціально для роботи з фреймворком .NET, проте саме поняття .NET дещо ширше. Якось Білл Гейтс сказав, що платформа .NET – це найкраще, що створила компанія Microsoft. Можливо, він мав рацію. Фреймворк .NET представляє потужну платформу для створення додатків. Можна виділити наступні її основні риси: Підтримка декількох мов. Основою платформи є загальномовне середовище виконання Common Language Runtime (CLR), завдяки чому .NET підтримує кілька мов: поряд з C # це також VB.NET, C ++, F #, а також різні діалекти інших мов, прив'язані до .NET, наприклад, Delphi. NET. При компіляції код на будь-якому з цих мов компілюється в збірку спільною мовою CIL (Common Intermediate Language) – свого роду асемблер платформи .NET. Тому ми можемо зробити окремі модулі однієї програми на окремих мовах. Кросплатформеність .NET є яку переносять платформою (з деякими обмеженнями). Наприклад, остання версія платформи на даний момент .NET Framework підтримується на більшості сучасних ОС Windows (Windows 10 / 8.1 / 8/7 / Vista). А завдяки проекту Mono можна створювати додатки, які будуть працювати і на інших

ОС сімейства Linux, в тому числі на мобільних платформах Android і iOS. Потужна бібліотека класів. .NET представляє єдину для всіх підтримуваних мов бібліотеку класів. І яке б додаток ми не збиралися писати на C# – текстовий редактор, чат або складний веб-сайт – так чи інакше ми задіємо бібліотеку класів .NET. Різноманітність технологій. Загальномовне середовище виконання CLR і базова бібліотека класів є основою для цілого стека технологій, які розробники можуть задіяти при побудові тих чи інших додатків. Наприклад, для роботи з базами даних в цьому стеку технологій призначена технологія ADO.NET. Для побудови графічних додатків з багатим насиченим інтерфейсом – технологія WPF. Для створення веб-сайтів – ASP.NET і т.д. Також ще слід відзначити таку особливість мови C # і фреймворка .NET, як автоматичне прибирання сміття. А це означає, що нам в більшості випадків не доведеться, на відміну від C ++, піклуватися про звільнення пам'яті. Вищезазначена загальномовного середовища CLR сама викличе збирач сміття і очистить пам'ять.

C# є об'єктно-орієнтованим і в цьому плані багато перейняв у Java і C++. Наприклад, C# підтримує поліморфізм, успадкування, перевантаження операторів, статичну типізацію. Об'єктно-орієнтований підхід дозволяє вирішити завдання з побудови великих, але в той же час гнучких, масштабованих і розширюваних додатків. І C# продовжує активно розвиватися, і з кожною новою версією з'являється все більше цікавих функциональностей, як, наприклад, лямбда, динамічне зв'язування, асинхронні методи і т.д.

Дана мова має широку колекцію існуючих контейнерів, що дають наступні можливість реалізації великої кількості контейнерів:

- IEnumerable
  - Загальна колекційна сутність, що має набір методів для проходження почергово по коленціях.

- ICollection
  - Колекція з більш широким набором методів, що забезпечують роботу та отримання об’єкту за індексом та отримання загальної кількості елементів як таких.
- IList
  - Колекція, що забезпечує роботу з набором даних, додавання, видалення та інші операції. Включає методи з проходження по колекції.
- IComparer
  - Тип даних колекції, що забезпечує роботу та підтримку порівняння у рамках роботи з даними.
- IDictionary
  - Типізована колекція;
  - Дані зберігаються в вигляді ключ-значення;
  - Може зберігати повністю серіалізовані об’єкти з інформаційним наповненням.
- IDictionaryEnumerator
  - визначає методи і властивості для лічильника словника.
- IEqualityComparer
  - визначає два методи Equals і GetHashCode, за допомогою яких два об'єкти порівнюються на предмет рівності.
- IStructuralComparer
  - Реалізує порівняння структур.
- IStructuralEquatable
  - Порівнняння структур;
  - Порівнняння значень всередені структур як посилань.

- ArrayList
  - Стандартна реалізація масивів.
- BitArray
  - Реалізація побітового масиву.
- Hashtable
  - Таблиця нетипізована, аналог словника.
- Queue
  - Структура даних, робота з якою йде за принципом FIFO.
- SortedList
  - Структура з характеристиками списку з вбудованою можливістю сортування.
- Stack
  - LIFO структура даних.

Перевагами даної мови є:

- out-змінні;
- зіставлення з шаблоном;
- шаблони і вираз switch;
- кортежі;
- розпакування кортежів (Деконструктор);
- локальні функції;
- покращення литералів;
- локальні змінні і повернені значення за посиланням;
- розширення списку типів, що повертаються асинхронними методами;
- throw вирази.

Компілятор виступає в якості сервісу статичних типів у простір імен фільтрів винятків await в блоках catch / finally ініціалізатора автовластивостей для читання, null-умовних операцій, інтерполяції рядків,

а також функцій, які є надзвичайно корисними під час розробки програмного забезпечення.

Далі наведено приклад коду на мові C#

```
// assembly: System.Drawing.dll
// assembly: System.Windows.Forms.dll
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsForms
{
    public class Program
    {
        [STAThread]
        public static void Main()
        {
            new MyPetrForm().ShowDialog();
        }
    }

    public class MyPetrForm: Form
    {
        Label label = new Label();

        public MyPetrForm ()
        {
            label.Text = "Init Part!";
            this.Controls.Add(label);
            this.StartPosition = FormStartPosition.CenterScreen;
            this.BackColor = Color.White;
            this.FormBorderStyle = FormBorderStyle.Fixed3D;
        }
    }
}
```

Лістинг 3.0.1 Приклад коду на мові C#

3.1.2 Near is a P2P

Дана бібліотеки P2P дозволяє:

- відкриття як Android NSD, хоча з більшою надійністю та простішим використанням NearDiscovery API;
- передачу серед клієнтів через простий у використанні API NearConnect.

NearDiscovery приймає ім'я хоста та налаштування у шаблоні конструктора для механізму виявлення. Об'єкт NearDiscovery дозволяє наступним відкриттям пов'язаних з оцінкою API:

- void make (String hostName);
- void makeNon ();
- void start ();
- void stop ();
- Set<Host> getAllAvailablePeers();
- boolean is ();
- boolean isDiscover ();

```
@NonNull
private NearDiscovery.Listener getNearDiscoveryListener() {
    return new NearDiscovery.Listener() {
        @Override
        public void onPeersUpdate(Set<Host> hosts) {
            // Handle updates of peer list here - some peer might have got removed if it wasn't reachable anymore
        }

        @Override
        public void onDiscoveryTimeout() {
            // This is called after the discovery timeout (specified in the builder) from starting discovery using
        }

        @Override
        public void onDiscoveryFailure(Throwable e) {
            // This is called if discovery could not be started
        }

        @Override
        public void onDiscoverableTimeout() {
            // This is called after the discoverable timeout (specified in the builder) from becoming discoverable
        }
    };
}
```

Лістинг 3.0.2 Приклад коду з бібліотеки Near is a P2P

NearDiscovery.Builder.setDiscoverablePingIntervalMillis () описує інтервал, за яким кожен клієнт передає інформацію про його існування. NearDiscovery.Listener.onPeersUpdate () отримує назву, навіть якщо одноранг вважається старим, тобто остання отримана трансляція була більш ніж удвічі більшою, ніж у ping-інтервалі.

```
allprojects {
    repositories {
        ...
        maven { url "https://jitpack.io" }
    }
}
```

Лістинг 3.0.3 Приклад підключення бібліотеки Near is a P2P

Розглянемо наступні обмеження даної бібліотеки.

- Передача файлів (реалізація) ще не тривіальна. Служби – це фон API, щоб отримувати сповіщення та розпочати їх на передньому плані, способи прослуховування публікації оновлень містяться в списку TODO.
- Поточний міні SDK становить 21. Знімаючи його, після тестування знову знаходиться в списку TODO.

```
dependencies {
    ...
    compile 'com.github.adroitandroid:Near:v1.1'
    ...
}
```

Лістинг 3.0.4 Приклад підключення бібліотеки через репозиторій Near is a P2P

3.2 Архітектура розробленого програмного забезпеченя

З огляду на область можливі області застосування даного алгоритму, було вирішено, що архітектура системи повинна бути побудована за клієнт-серверною моделлю модульного типу. Модульна архітектура дозволяє реалізувати принцип DependensyInjection, що з точки зору закладення архітектури може в рамках розробки на компіляційних мов програмування з строгою типізацію надати результуючому програмному забезпеченню гнучкості та можливості підтримки реалізації поліморфічних хуків.

Гнучка поведінка системи може диктуватись трьома видами реалізаціях поліморфізму, а саме:

- Статичному:
    - реалізація за допомогою перевантаження методів існуючої функціональності;
    - зміна поведінки забезпечується зміною методів викликів.
- Динамічному:
    - зміна стану відбувається за допомогою зміни параметрів, що приходять в тіло функції або методу, за допомогою внутрішніх операцій та за допомогою роботи зі вказівками.
- Параметричному:
    - Зміна стану відбувається на основі зміни виконуваного типу, методи, що асоціюють з одним типом, наприклад, рядками, працюють по-іншому з іншими. Зазвичай такі прийоми застосовують до групам класів з однаковими інтерфейсами, що наслідують від одного абстрактного класу або інтерфейсу.

Фізична архітектура програмного забезпечення розділена на частини: клієнтське відображення, компілятор зв'язки елементів системи, внутрішня логіка з системами використання внутрішніх пакетів та зовнішні оптимізаційні рішення об'єднані в систему покращення продуктивності.

Елементи фізичної схеми описані далі в тексті та зображена на рисунку 3.4:

- Клієнтська частина – Web App
    - XML – мова розмітки з багатою семантикою, що використовується для опису відображення екранів програмного засобу, екрану мобільного додатку, що розширюється на екрані смартфона.
    - Resources – ресурси додатку, що описують кнопки, назви табів, назви фонів, текстові константи, розширення, внутрішні зображення, що не можуть постійно бути вивантажені в пам’ять телефону, використані та ініціалізовані за типом scoped з dispose елементів після зникнення необхідності в останніх.

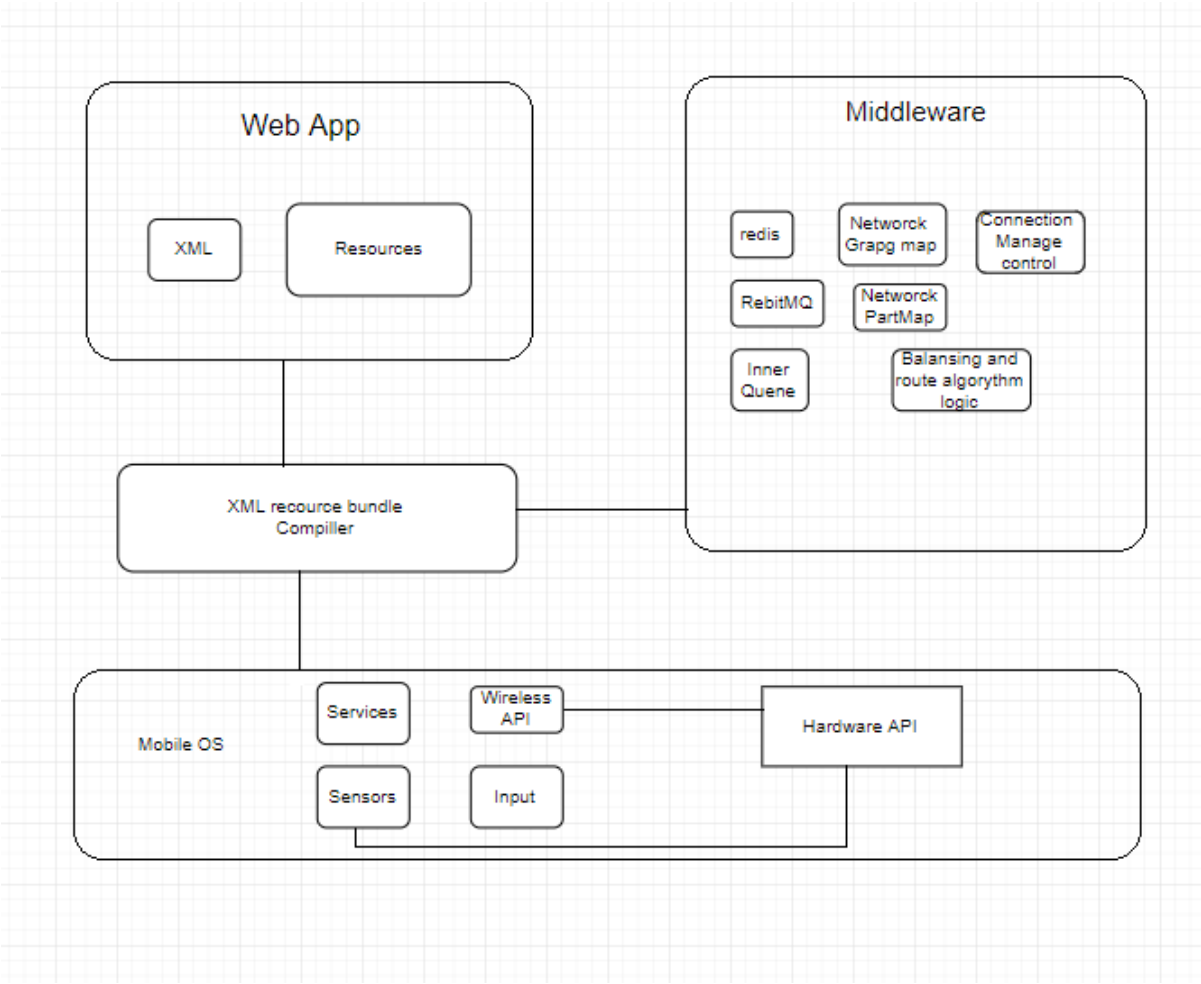


Рис. 3.4 Фізична архітектура мобільного додатку

- Middleware – внутрішня логіка роботи застосунку, об’єднана з модулями розширеннями, базами даних, внутрішніх станів системи(StateMachine).
    - Redis – внутрішня база даних додатку, якій притаманні надзвичайно висока продуктивність зі збереження та обробки даних. Документо-орієнтована система збереження даних типу ключ – значення на основі словникового відображення, з відкритим кодом, що можна вільно забрати та імплементувати в сам додаток. Використовується для оптимізації заголовків та в якості нової структури даних замість векторного представлення стандарних структур даних в AODV алгоритмі, на зміну двохзв'язного списку структур вершин зі повним сканом – пошуком по ньому.
    - RebitMQ – внутрішній програмний компонент, орієнтований на обробку черги повідомлень. Використовується для нумерування пакету, обробки черги, втунтрішнього розфарбовування графу в рамках взаємодії-передачі повідомлення.
    - Inner Quene – хешована таблиця даних повідомлень, що зберігає загальну колекцію повідомлень, що приходять в вузол та знаходяться там доти, поки не відбудеться відправка пакету.
    - Networck GrapmMap – загальна згенерована мапа графу мережі, вершини та ребра з їх характеристиками.
    - Networck PartMap – часткова мапа мережі сусідніх вершин, під’єднаних до поточнох вершини, використовується для оптимізації проміжних оновлень таблиць.

- Connection Manage control – модуль програмного застосунку, що відповідає за управління підключеннями до данного вузлу.
- Balansing and route Algorytm – реалізація алгоритму, імплементована в додаток.

- MobileOS – API, що надає мобільний додаток для роботи з embedded системами та можливостями отримання даних від них для оптимальної роботи системи та більшо повного розуміння стану, в якому знаходиться сама система з фізичної точки зору, а саме:
  - Services – сервіси, що задані та створені у рамках функціонування самої операційної системи Android, інтерфейси до даних та зв'язки з ними.
  - Sensors – набір вбудованих API для роботи з сенсорами апарату на основі ОС Android, набір останніх залежить від, самого апарату.
  - Wireless – набір драйверів, що дозволяє керувати бездротовим підключенням до інших пристроїв.
  - Input –набір драйверів, що зчитують всі користувацькі вжаємодії з системою та апаратом та надають доступ до клавіатури та зчитування даних з монітору, мікрофону, гіроскопу тощо. Переплетений з сенсорами.
  - HardwareAPI – набір викликів системних та прикладних утиліт, що дозволяють взаємодіяти з апаратною частиною системи, створювати додаткові демони всередині машини Linux.

Дана реалізація має наступні переваги та покращення.

- Система не прив'язана до певного алгоритму, алгоритм через механіхм dependencyInjectiuon може змінюватися на інший, впровадження новго алгоритму набагато простіше.

- Логіка роботи з самою системою не покладена на реалізацію покращеного алгоритму.
- Відв’язаність дозволяє переностити та імплементувати сам алгоритм, відмінний від додатку, на інші системи.
- Рівень представлення даних являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і отримання від нього керуючих команд та даних.
- Прикладний рівень реалізує основну логіку застосунку, на якому здійснюється необхідна обробка інформації;
- Рівень доступу до даних забезпечує зберігання даних та доступ до них.

3.3 Особливості реалізації алггоритму

3.3.1 Аналіз задач до розробки

Як зазначено в розділі 2, цільовим алгоритмом безперервної передачі даних для дослідження та покращення є AODV, причому дану модифікацію необхідно додати і для умови використання в умовах швидкого розгортання мережі.

Модифікація полягає у наступному.

- Виправити проблему зі зниканням вузлів та зациклюванням повідомлення всередині мережі.
- Модифікувати передачу повідомлення по ребрам агентів мережі що є довіреними, та перебудовувати поточну таблицю.
- Замінити структури даних за замовчуванням на відповідні до динамічної зміни характеристик.
- Зменшити час затримки повідомлення.
- Зменшити кількість втрачених пакетів для підтримки безперервного обміну інформацією.
- Зменшити кількість трафіку, що генерується мережею під час роботи.

Вище зазначені проблеми є критичними та вирішуються наступними алгоритмічними впровадженнями.

3.3.2 Проблема зникнення вузлів

Проблема зникання вузлів є однією з найважливіших для оптимізації в рамках даної роботи та розробки програмного забезпечення для алгоритмів маршрутизації. Вирішення даної проблеми може набагато покращити загальний час кожного вузла при певних навантаженнях.

При побудованому шляху мережа приймає рішення про відправку групи пакетів по перевіреному каналу, але через те, що природа динамічних каналів непередбачувана, іноді вузли такого роду можуть зникнути з поля зору мереж, при чому відправлені пакети починають шукати свій пункт призначення поза мажами налаштованих маршрутів і «блукати» по мережі, що призводить до зациклювань та втрати через певну кількість стрибків, занесених в хедери повідомлення як неактивних, після чого ініціюється перебудова нового маршруту та повторне надсилання повідомлень, що зменшує швидкодію та призводить до втрати часу та загальної продуктивності мережі.

Рішенням даної проблеми буде вводження проміжної додаткової таблиці маршрутизації зі станами сусідів, де глибина $n < 3$, тобто сусід знає ще про сусідів вузла ініціатора надавання маршруту.

Структура даних, що підходить для цього, описана в 3.3.4 та після обміну та відсутності в подальшому шляху передавача йде перемикання на нову маршрутизацію.

Варто зазначити, що через малу глибину обізнаності в топології мережі оновлення таких таблиць буде проходити набагато швидше, і в рамках роботи загального дистанційно-векторного алгоритму, набагато частіше генеруватиметься менше викликів та менше трафіку для часткових синхронізації вузлів.

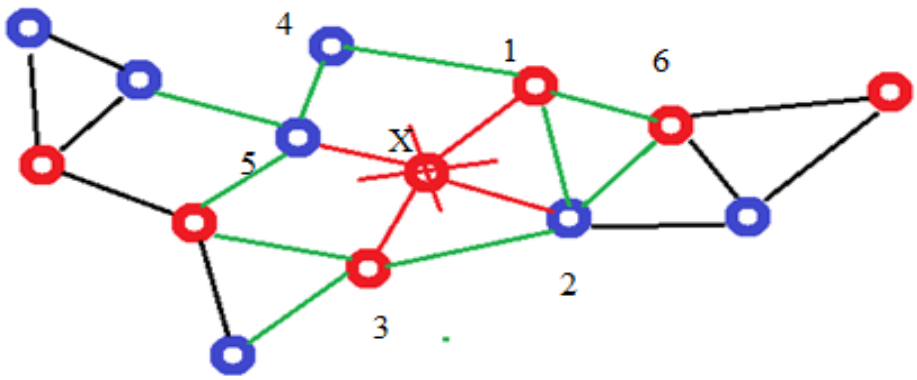


Рис.3.6. Частина графу мережі з побудованим шляхом руху трафіку.

На рисунку 3.6 зображено приклад роботи модифікованого алгоритму, де:

- кола – вузли мережі;
- червоні кола передають потік пакетів з найкращим шляхом;
- перекреслене коло – вузол, що буде прибраний;
- червоні ребра – 1-й рівень обізнаності топології мережі, що зберігається у вузлі X;
- зелені ребра – 2-й рівень обізнаності мережі, що зберігається у вузлі X.

У випадку зникнення вузла X у проміжних картах відбудеться перевірка на наявність вузлу в сусідах (лістинг 3.1).

```
(this.target(source.node).contains(x =>
x.super(x.contains(x.mappedNodeUniqeId)).getMonade as
expression)
.Invoke(this.localGraphMap => x.reduce((node, target) =>
(x.getAnyClosest(oldWay).id == x.nextNodeId)))
```

Лістинг 3.1 Перевірка наявності в графі шуканого вузла

Дане покращення надає можливість в динамічному режимі завдяки асинхронності роботи алгоритму без втрат швидкодії, знаходити шлях назад до відправника, надаючи йому відповідь на виклик надсилання нового пакету, модифікувати передачу повідомлення по ребрам агентів мережі. що є довіреними, та перебудовувати поточну таблицю.

Під час передачі задля усунення зациклювання пакетів в середині графу необхідно додати наступну модифікацію.

- Під час розсилання в ширину повідомлення індексувати кожний пакет відповідно до комбінацій номеру машини, вузла, номеру повідомлення, часу створення задля унікальності повідомлення.
- При отриманні повідомлення вершиною виконувати «розфарбовування» вузлу відносно повідомлення, позначаючи його як вже отримане, зберігаючи хеш тіла повідомлення, задля усунення повторного зациклювання (рис 3.7).
    - Зберігається тільки хеш, займає мало місця.
    - Порівняння за хешем є швидкою операцією в реалізації Redis.
- При швидшому оповіщенні шляху 1-2-3 ніж 1-3 запит на відправку пакету, шлях 3-1 буде заблокований.

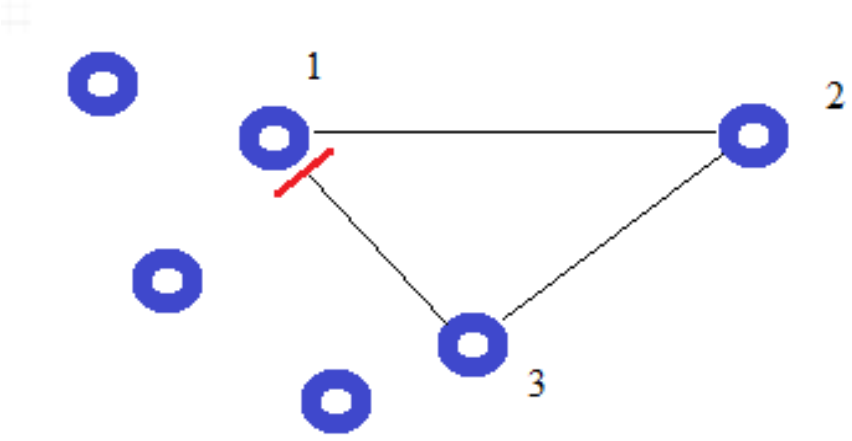


Рис 3.7 Розфарбовування вершини графу

Дане покращення разом з хешуванням заголовків алгоритму HPACK є найбільш продуктивним завдяки уникненню зациклювань.

Стискання заголовків також знижує кількість службово го трафіку що позитивно впливає на загальну тенденцію роботи алгоритму.

3.3.3 Зменшення часу затримки повідомлення

В якості покращення можемо змінити час затримки повідомлення під час обробки повідомленневого пакету в рамках вузла за допомогою оптимізації бази даних.

На даним момент дистанційно-векторний алгоритм базується на векторі значень маршрутизаційної таблиці, що є двохзв'язним списком (рис. 3.8). Пошук проходить за допомогою сканування всієї колекції такого роду записів в таблиці, що на великій кількості повідомлень та вузлів дає складність *n* (довжина вектору).

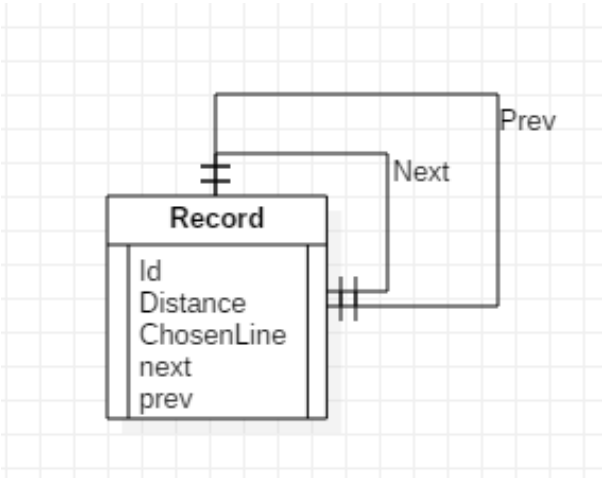


Рис. 3.8 Поточна схема збереження таблиці маршрутизації

Застосовуючи графо-орієнтовний підхід в комбінації з реалізацією redis хеш таблиць та колекцій і винісши її в оперативну пам'ять, отримуємо приріст в (log(n)- $\sqrt{n}\log(n)$ разів за рахунок оптимізації таблиць зі схемою, наведеною на рис. 3.9.

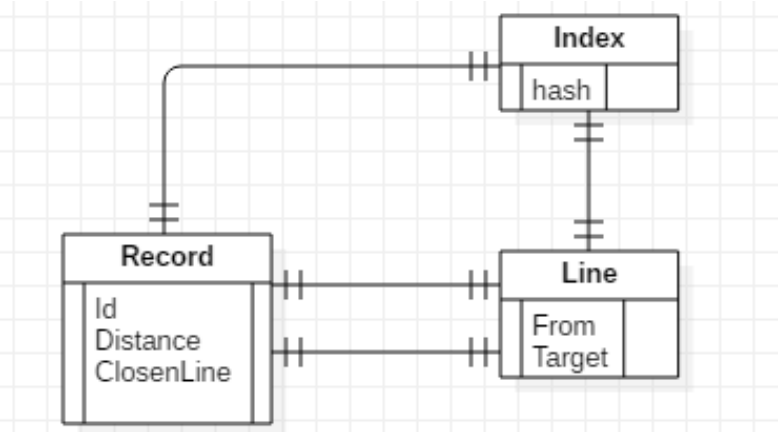


Рис. 3.9 Схема збереження таблиці маршрутизації

Зменшуючи сутності, ми зменшуємо частоту оновлення таблиці, відповідно, створюємо можливості для створення індексу без втрати на оновлення в рамках роботи redis. Надаючи доступ зі зв'язком один до одного можна, не хвилюватись про продуктивність. Приріст до логарифмічної величини забезпечується індексуванням, де таблиця Line є одним ключем з унікальними значеннями.

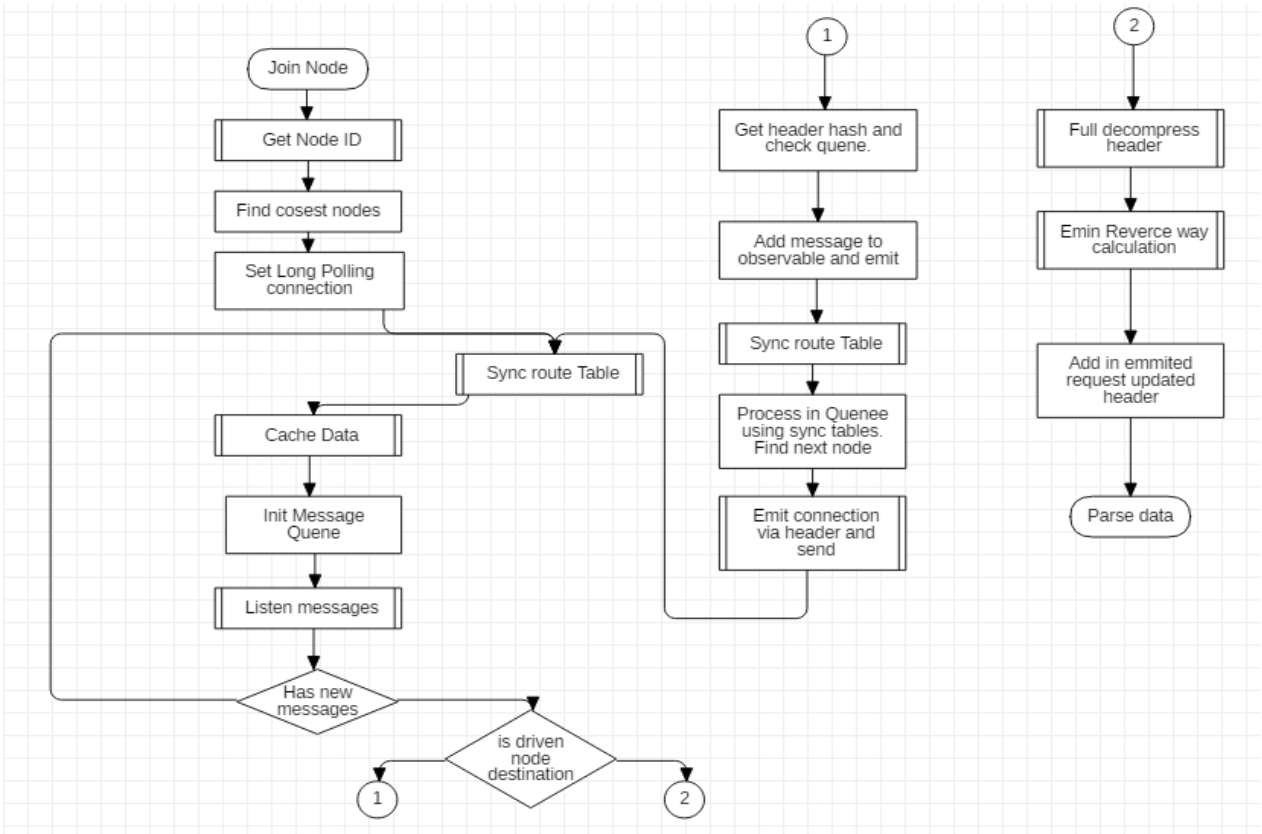


Рис 3.4 Рисунок роботи алгоритму.

3.3.4 Впровадження модифікацій

Після впровадження вище зазначених модифікацій спостерігаються достатньо великі прирісти в продуктивності застосунку. Детальніше це описано в розділі 4.

3.3.5 Особливості програмної реалізації застосунку

Програмний застосунок розроблений на основі програмної та апаратної платформи Andoid, що дозволяє створювати додатки для великого кількості пристроїв.

Розроблене програмне забезпечення дозволяє обмінюватись повідомленнями за допомогою встановлення однорангового з'єднання між декількома пристроями через бездротовий інтерфейс.

В ході розробки необхідно було обійти потребу постійного підтвердження передавання інформаційних повідомлень без дозволу, тому для використання даного продукта необхідно надавати root-права на телефоні поточному користувачеві, що є доволі ризиковою операцією, але надає безліч можливостей та знімає багато обмежень з штатного режиму взаємодії з телефоном.

- Програмний додаток дозволяє встановлювати довгострокове з’єднання між вузлами.
- Надсилати повідомлення як по звичайному каналу, так і по захищеному.
- Запам'ятовувати останні вузли та повторно до них приєднуватись.
- Відслідковувати внутрішній стан роботи додатку.

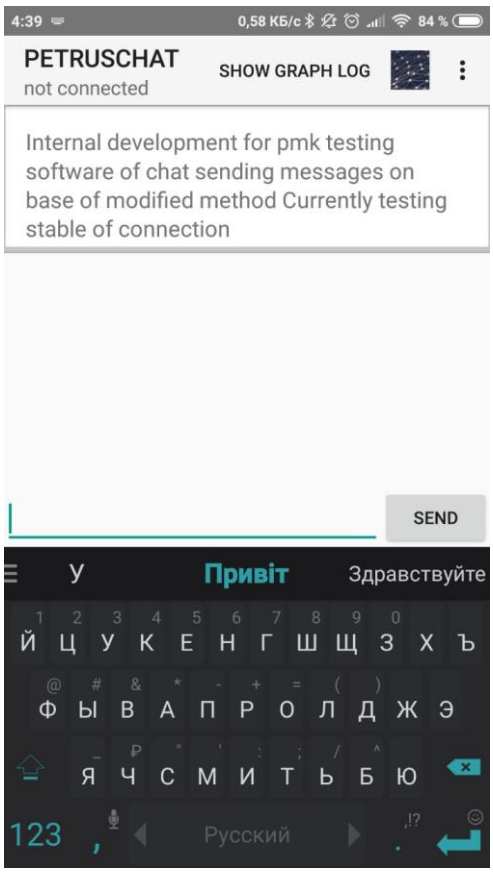


Рис. 3.10 Головне меню розробленого додатку.

3.4 Висновки

У рамках дослідження можливостей модифікації алгоритмів безперервної передачі даних та швидкого розгортання мережі було розроблено покращення щодо оптимізації критичних моментів у MANET-мережах.

Розроблений програмний додаток вміщує модифікований алгоритм з можливостями взаємодії з апаратним API для системи Android та підтримує обмін інформації в мережі.

Досліджено та запропоновано програмні способи вирішення виділених проблем та усунуто проблему з зациклюванням за допомогою зафарбовуванням вершин графу в мережі.

РОЗДІЛ 4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1. Методика оцінювання ефективності методів безперервної передачі даних

Для оцінки якості роботи методу безперервної передачі даних необхідно дослідити на визначити ключові параметри, що є важливими для роботи самого алгоритму як в мінімальному режимі, так і в максимальному.

Після детального вивчення питання можна ввести наступні показники:

- кількість вузлів, які беруть участь в маршрутизації;
- наявність у кожного вузла маршруту до будь-якого іншого вузла в мережі;
- можливість виникнення петель;
- рівень знання кожним вузлом топології всієї мережі;
- кількість ресурсів для знаходження резервного каналу зв’язку;
- використання службового трафіку.

Також можна ввести наступні кількісні характеристики:

- кількість вузлів;
- час на подолання шляху;
- затримка у вузлі.

Дані характеристики максимально повно оцінюють покращення в рамках безперервної передачі даних в децентралізованих мережах, зі змінною природою.

4.2 Результати роботи модифікованого методу

Тестування було проведено на комп'ютерній моделі, що імітувала роботу системи. На рис. 4.1. граф має 12 вершин та приблизно однакову довжину ребер. Ребра генерувались відповідно до положення вершин, котрі, в свою чергу, розставлялись випадково.

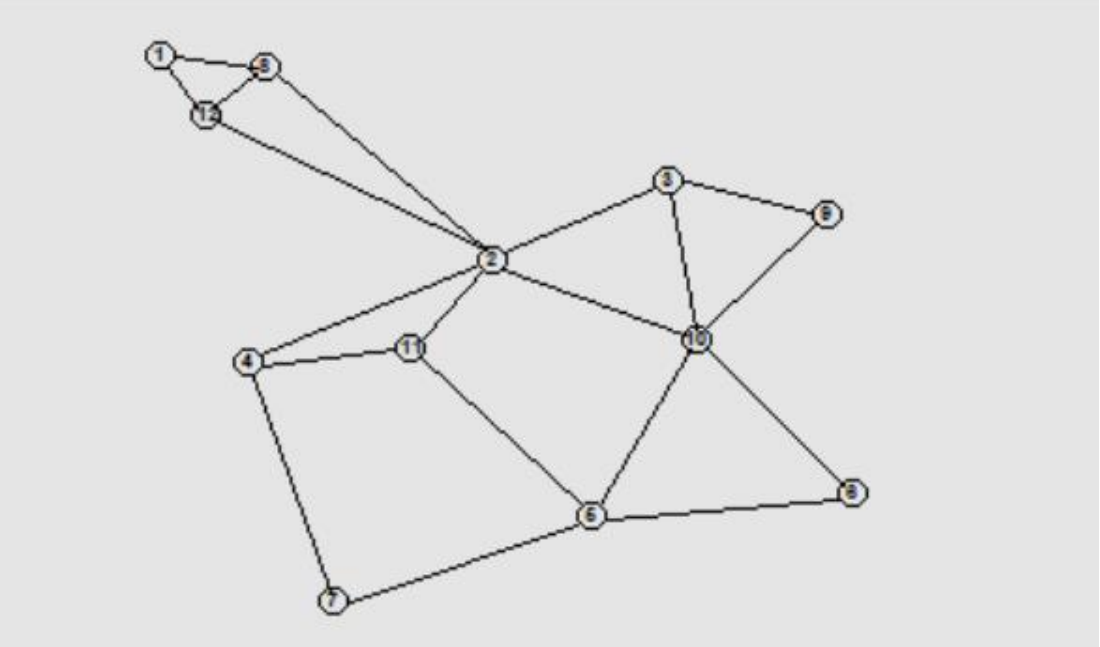


Рис 4.1 Досліджуваний граф мережі

У таблиці 4.1 відображено результати тестування та наведено результати порівняння методів.

Загальний результат є середнім всіх спроб надсилання пакету від одного випадкового вузла до іншого в протилежному місці з проведеням відповідної ініціалізації.

Таблиця 4.1

Порівняння загальних характеристик алгоритмів

Метод з модифікацією	Метод без модифікації	Характеристика
7.31	6.15	кількість вузлів, які займаються маршрутизацією
+	+	наявність у кожного вузла маршруту до будь-якого іншого вузла в мережі
0.2	0.4	можливість виникнення петель
+	+	рівень знання кожним вузлом топології всієї мережі
70%(Хор) 84%(Пог)	76%(Хор) 82%(Пог)	кількість ресурсів для знаходження резервного каналу зв’язку
352b	438b	використання службового трафіку

Таблиця 4.2

Часова затримка обробки в кожному вузлі

Навантаження kb\s	Модифікований (затримка (с))	Не Модифікований (затримка (с))
20	0.0093	**0.0081**
40	**0.00103**	0.00119
50	**0.0142**	0.0157
70	**0.0162**	0.0171
100	**0.0271**	0.0276
120	0.0302	**0.0294**

Результати, наведені в таблиці 4.2, можуть дати чітке бачення того, що на інтервалі від 40 до 100 kb\s навантаження на мережу, запропонований алгоритм також має ряд покращень, а саме зменшення часу затримки обробки повідомлення у вузлі, що замірювався як середнє статистичне даного тестування. Дане покращення дозволяє бачити що:

- затримка в рамках вузла зменшилась на 7-15 %, в залежності від топології вершини графу та взаємодії агенту мережі з іншими агентами;
- у рамках роботи мережі при кількості вузлів, що буде зростати, дане покарщення може збільшити продуктивність мережі, а саме її пропускну здатність.
- час шляху проходження повідомлення, що залежить від суми проходження та обробки повідомлення всіми вузлами-передавачами мережі, також покращився (залежить від положення графу у просторі та ребер).

4.3 Висновки

У даному розділі проведено аналіз розробленого методу безперервної передачі даних в однорангових децентралізованих мережах. Метод дозволяє розв’язати поставлену задачу, даючи змогу створити рухомі бездротові мережі на основі MANET.

Проведено аналіз якості методів маршрутизації трафіку в середині мережі, що в даному випадку зводиться до оптимізації маршрутизації на основі Ad hoc On-Demand Distance Vector алгоритму. Було виділено 4 основні показники: швидкість встановлення шляху, затримка повідомлення в одному вузлі, виникнення зациклень, кількість службового трафіку.

Крім того, для тестування було змодельовано граф, що імітує бездротову однорангову меержу, з дванадцяти вершин. Тестування проводилось в кількості більше ста ітерацій. При кожній ітерації

випадковим чином обирались вершини та було ініційовано передачу даних між кожною вершиною.

Завдяки використанню створеного програмного забезпечення було проведено аналіз показників якості безперервної передачі даних та маршрутизації мережі запропонованим методом. Запропонований метод показав кращі на 5-10% результати зменшення часу затримки в рамках вузла та при обробці повідомлення. Зменшено кількість службового трафіку на 15%. При чому зросли витрати пам’яті, що не є важливим показником в оцінюванні даного алгоритму.

РОЗДІЛ 5 ПОБУДОВА БІЗНЕС МОДЕЛІ

5.1. Опис проблеми та дерево проблем

5.1.1. Анотація проекту

Проект спрямований на розробку та тестування способу знаходження альтернативних засобів зв'язку в умовах зникаючого сигналу. Розробка передбачає вирішення проблеми зникнення сигналу в умовах білих плям. Тестування передбачає створення симуляції мережі з декількох пристроїв (мінімум два), поміж яких буде відбуватись передача інформації за допомогою розробленої бібліотеки.

5.1.2. Опис проблеми, на розв'язання якої спрямований проект

Розробка способу знаходження альтернативних каналів зв'язку в умовах слабкого сигналу полягає в знаходженні та розробці програмного засобу, що за допомогою апаратних можливостей цільової платформи зможе в режимі реального часу аналізувати всі наявні засоби зв'язку та використовувати ті, котрі будуть їй доцільні задля створення зв'язку в разі відсутності цільової мережі або проблем з передачею інформації через неї.

В сучасному світі користувачі мереж дуже вибагливі до наявності підключення до хоч однієї мережі та доступу до загальнодоступних ресурсів, через силу певних обставин, користувач як фізична особа так і підприємство в цілому може відчувати дискомфорт під час користування мережею на крайових зонах, або під час пошкодження ліній зв'язку чи навіть просто слабкого сигналу. Це може призвести до втрати переданих пакетів, помилок в роботі сучасних високорівневих додатків, що втрачають зв'язок з основною мережею та додатків, що базуються на десктопі в рухомих засобах або помилок в роботі статичних додатків з клієнт-серверною архітектурою.

Основна спрямованість проекту – винайдення способу, що використовуватиме альтернативні шляхи отримання доступу до певних

мереж, покращуватиме якість зв’язку для невеликих груп пристроїв, та зменшить кількість втрачених пакетів шляхом агрегації та покращення загальної пропускної можливості мережі. Використання даного способу покращить досвід користувача при роботі з додатками, яким потрібний Інтернет та сферам бізнесу, які використовують підключення до мережі для постійного моніторингу загального становища.

5.1.3 Мета та завдання проекту відповідно до проблеми

Метою проекту є винайдення та тестування способу організації бездротових мереж. Для цього необхідно виділити вже існуючі методи підвищення ефективності мереж, виявити їх недоліки та переваги та обрати шлях розвитку проекту відповідно до виявлених переваг та недоліків. Рішення дозволить прискорити роботу логістичних центрів та дата-центрів, прискорити обробку інформації, що дозволить надати проекту подальшу його оптимізацію. Також метою є створення симулятору для отримання статистичної інформації для порівняння з аналогами винайденого способу. Наступними кроком є презентування отриманих результатів та знаходження інвестицій для встановлення або перевстановлення подібних систем або заміна старіших, менш оптимальних. Залучення сторонніх розробників методом надання API у користування розробникам не для комерційних цілей дасть можливість використовувати та розвивати базу коду та функціональності реалізованого способу.

В основі методу є використання апаратних можливостей мобільних пристроїв, що мають доступ до платформи, можуть взаємодіяти з мобільними мережами, мережами wifi, 3G та LTE, та мають програмне забезпечення, що контролює кореляцію якості зв’язку та величину пропускного сигналу відносно вибраної групи засобів бездротового зв’язку. Пристрої, що мають в собі програмне забезпечення, що імплементувало в себе дану бібліотеку з рішеннями, можуть комунікувати один з одним на основі мереж, що об’єднують пристрої один з одним на основі long-pooling рішень.

5.1.4 Дерево проблем.

Базове дерево проблем проекту зображено на рисунку 5.1.

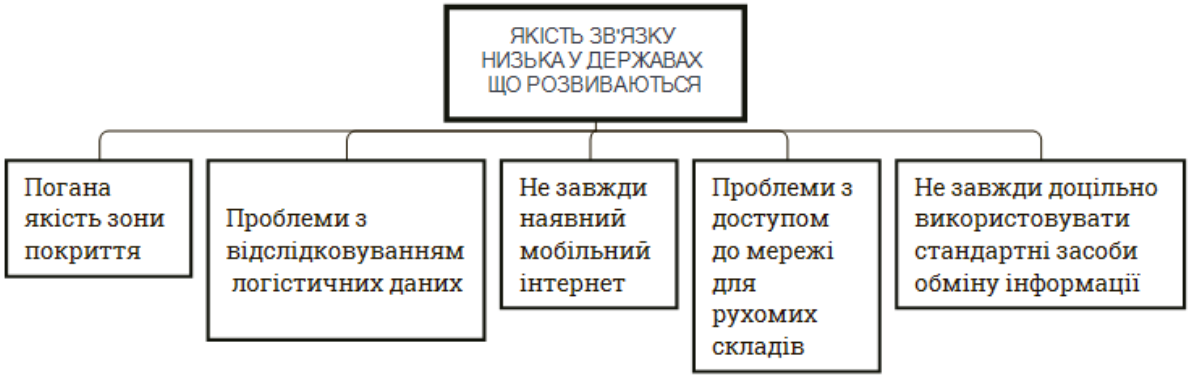


Рис. 5.1. Дерево проблем проекту

Найголовніша проблема для сучасного ринку – це наявність у державах, що розвиваються, пристроїв, що мають технічне забезпечення високого рівня як для розвинутих держав, але відсутність зв’язку того рівня, що можуть розкрити його можливості, а саме:

- високу швидкість підключення до мережі;
- синхронізацію з хмарними сервісами;
- агрегацію даних постійно та збереження та обробку їх віддалено;
- поширення даних між іншими пристроями, прив’язаними до певного користувача.

Зі сторони бізнесу це виливається в конкуренцію і величезні витрати на конкурування з закордонними сервісами, або взагалі неспроможність конкурувати з закордонними фірмами (Azure) в рамках низько-швидкісної передачі даних.

Загальна проблема спричиняє неспроможність покрити вимоги користувача та викликає проблеми з конкуренцією та внутрішні проблеми

на ринку телекомунікацій та логістики, котрим важливо отримувати дані в реальному часі.

Дана проблема загальна, але вона вміщує ще декілька підпроблем, кожна з яких є комплексною та, може бути розбита на менші частини. Дане розбиття відображає актуальність проблеми як в наукових цілях, так і в інтересах ринку. Загальну проблему можна розбити на наступні:

- Погана якість зони покриття – на даний момент, згідно з даними за 01.02.2018 р., велика кількість маршрутів не мають зони покриття взагалі, причому така проблема є у всіх операторів. Дивлячись на рис. 5.2 та 5.3 можна побачити, що багато районів з транспортними маршрутами мають так звані «сліпі зони». Це відбувається в основному через фізичну відсутність ретрансляторів, що підтримують даний зв’язок.
    a. Відсутнє технічне забезпечення – під технічним забезпеченням мається на увазі відсутність ретрансляторів передачі, або їх низька щільність на певній території, що спричиняє втрату сигналу або неспроможність мережі підтримувати пристрої.
    b. Санітарно-технічні заборони на встановлення обладнання – в українському законодавстві наявні парні обмеження поряд з певною територією або просто через загальний ефір встановлювати ретранслятори.
    c. Незацікавленість бізнесу – монополісти ринку можуть роками не оновлювати обладнання, маючи клієнтську базу, та спокійно отримувати прибуток. З точки зору економіки, будь-які зміни або затрати не вигідні.
    d. Відсутність фінансування: деякі території або регіони не мають достатнього фінансування для підключення додаткових апаратних потужностей.

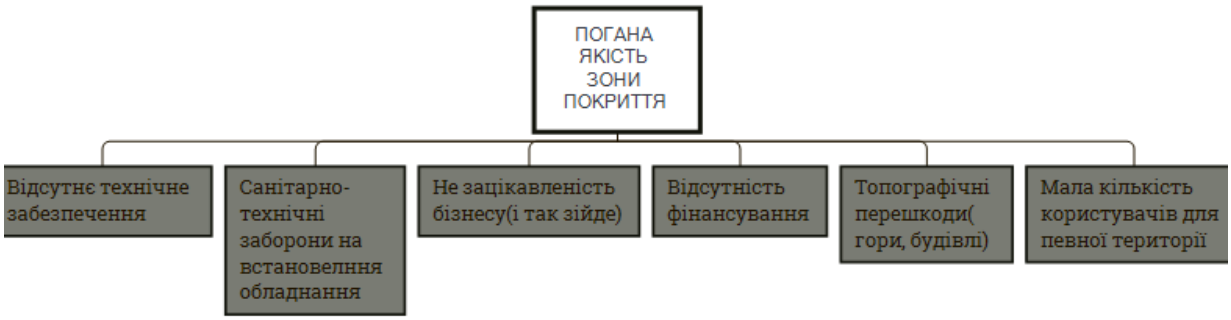


Рис. 5.2. Дерево проблем, деталізація 1

e. Топографічні перешкоди (гори, будівлі) – іноді сигнал навіть від великої кількості ретрансляторів достатньої потужності для нормальних умов може бути спотворений територіальними особливостями.

f. Мала кількість користувачів для певної території.

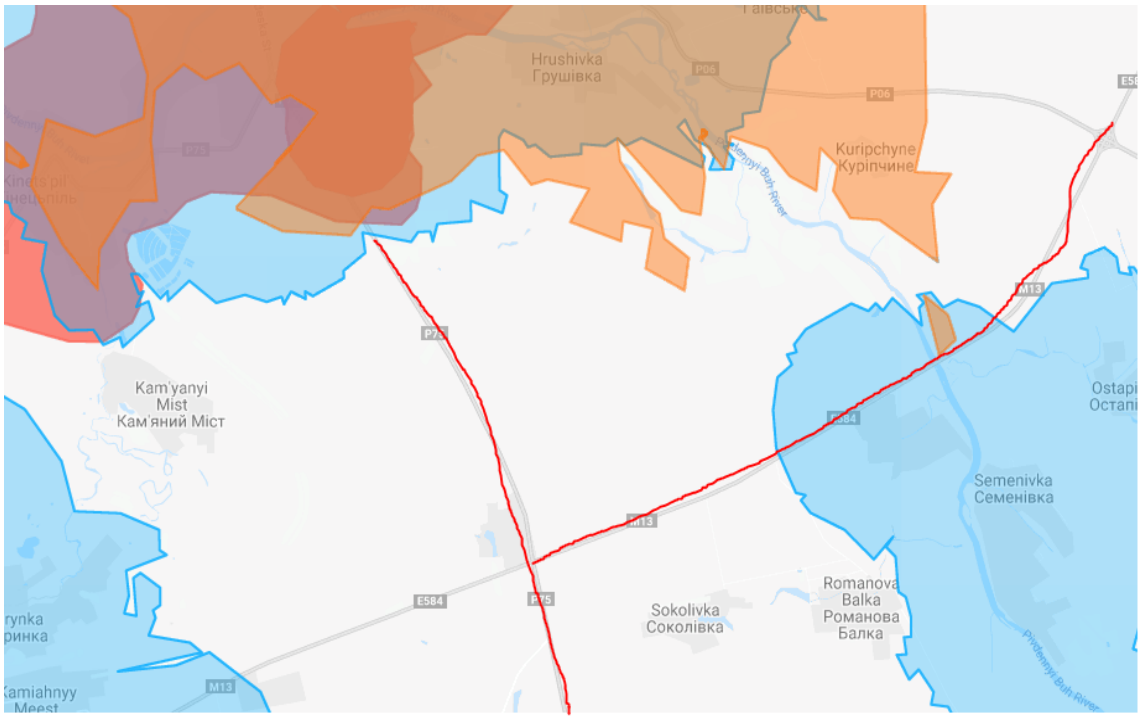


Рис. 5.3. Приклад карти покриття

- Проблеми з відслідковуванням логістичних даних – логістичні дані є важливими для продуктивного бізнесу в сфері телекомунікацій та логістики у транспортних системах. Через погану якість зв’язку дані

трекінгу та трасування можуть бути втрачені. Відслідковування транспорту в реальному часі, його координація, обмін вантажів, є пріоритетною задачею, що базується на якості збору та передачі даних про місце перебування.

- Слабке технічне забезпечення – при навантаженні на мережу одночасно великої кількості запитів і реальному часі мережа може виходити з ладу, що є неприпустимим.
- Слабке програмно-апаратне забезпечення – програмна денормалізація запитів і розбиття на асинхронні задачі можуть покращити продуктивність, але в цілому такого роду програмне забезпечення дороге для замовника та важко підтримуване.

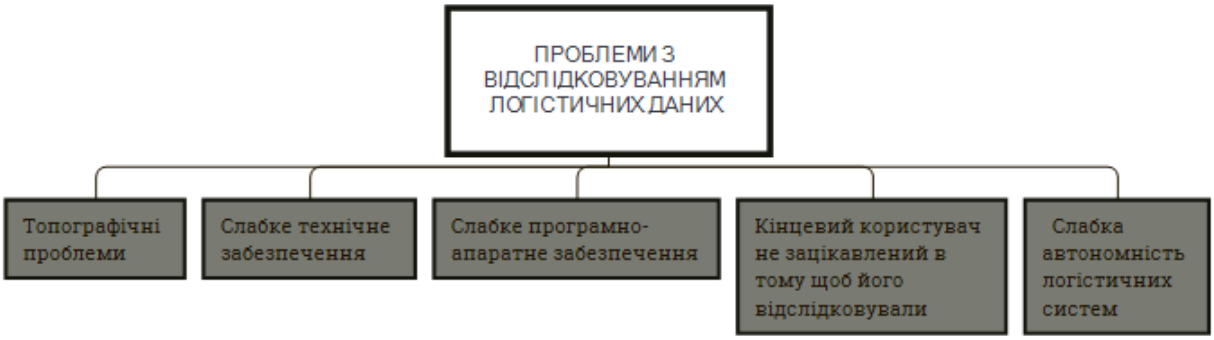


Рис. 5.4. Дерево проблем, деталізація 2

- Кінцевий користувач не зацікавлений в тому, щоб його відслідковували – інформація про поточне місце положення не завжди добровільно може бути отримана від користувача, і не завжди у користувача є довіра до певного сервісу, що використовується.
- Слабка автономність логістичних систем автономність є ключовою складовою успішної системи. Важливо підтримувати незалежність будь-якої однієї з керованих величин від змін інших керованих величин.

- Не завжди наявний мобільний інтернет (відсутні точки підключення, слабкий сигнал, перевантажена мережа, блокування зі сторони провайдеру). Мобільний інтернет можуть використовувати як мобільні, так і стаціонарні пристрої, адже в такому випадку знижуються витрати на прокладання фізичного доступу через кабелі. На даний момент доступ до мобільного інтернету можуть забезпечувати наступні стільникові стандарти: GSM, UMTS, LTE. Дані стандарти з'єднують користувача з загальною мережею всіх користувачів та забезпечують обмін інформацією. Цей критерій дуже важливий як і для фізичних осіб-користувачів, так і для підприємств, що використовують мобільні рішення.

Рис. 5.5. Дерево проблем, деталізація 3

- Проблеми з доступом до мережі для рухомих засобів: під час швидкого руху (поїздів, враховуючи вище згадані проблеми з апаратними обмеженнями, можливі проблеми зі зв'язком, а саме зникання, збільшення часу відповіді тощо. Висока швидкість перемикання станцій знижує швидкість доступу, дороги віддалені від точок передачі сигналу, можливі втрати пакетів через перемикання станцій.

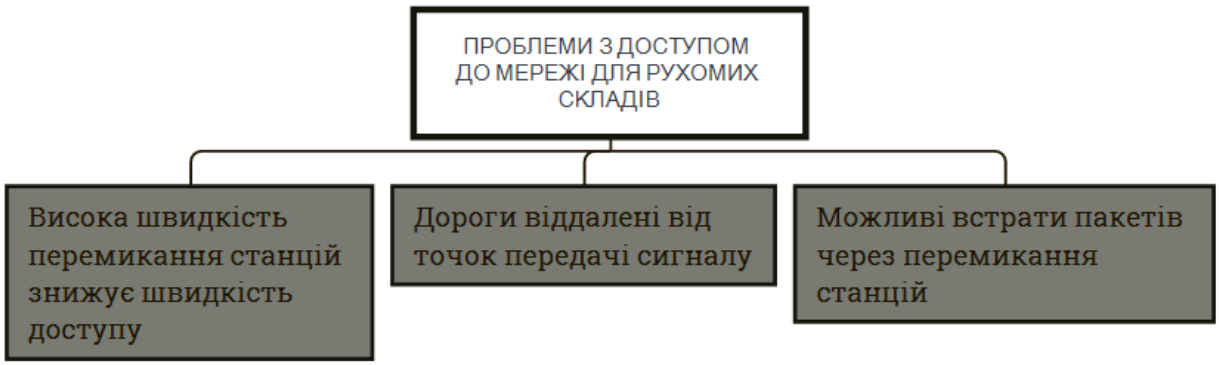


Рис. 5.5. Дерево проблем, деталізація 4

- Не завжди доцільно використовувати стандартні засоби обміну інформації – зазвичай більшість інформації проходить дротовими або бездротовими маршрутами. Не завжди доцільно використовувати мобільний інтернет з посередником в якості передавача (провайдеру мобільного інтернету), для того, щоб підправити повідомлення, такий трафік може бути відслідкованим та скомпрометованим.

5.2. Опис бізнес моделі продуку

5.2.1 Зацікавлені сторони проекту.

Зацікавлені сторони можна умовно поділити на такі групи: компанії, звичайні користувачі, міні сервіси (види підприємств або автоматизованих програмних комплексів, у котрих монетизація відбувається за допомогою вбудовування в інші види бізнесу на основі IT рішень), та дослідницькі центри.

1. Компанії.

Тут зібрані особи, що напряму зацікавлені у якісному продукті або рішенні та зацікавлені в розробленні добре діючого та продуктивного прототипу, способу або рішення, що може бути імплементоване для

збільшення прибутку. На перших етапах компанії не зацікавлені в інвестуванні, але в подальшому можуть стати доволі потужним гравцем на ринку під час розвитку даного продукту. Це всі, хто зацікавлений саме у отриманні доступу до використання методу, тобто «цільова аудиторія» методу. До неї входять усі фірми, підприємства, корпорації, заклади, що працюють з інформацією у тому чи іншому вигляді, а також можуть бути носіями статистичних даних або тестовими платформами методу.

2. Звичайні користувачі

Звичайні користувачі займають ключову роль в розвитку додатку, кожен користувач з мобільним пристроєм може поширювати дану програму як рекламу, так і додатки, що можуть використовувати даний спосіб для організації однорангових мереж. Досвід користувачів набагато більший за той, що може надати компанія. Фінансування від мікротранзакцій забезпечує стабільний заробіток для користувачів даної технології та покриває вимоги користувачів щодо вимог до якості зв’язку.

3. Науково-дослідницькі центри.

Центри, що зацікавленні в досліджені даної технології і досліджують її в суміжних рішеннях або досліджують рішення, пов’язані за схожою проблематикою. Для них частково важливий вміст в роботі теми та тематики мало досліджуваної попередньо, наукова новизна, наявність документації відповідно вимогами, консультуванням з розробниками.

4. Міні-сервіси.

Всі види послуг і сервісу, які надаються за допомогою електронних засобів (наприклад, за допомогою Інтернету). Прикладами можуть бути: інтерактивні банківські (Інтернет-банкінг) та фінансові послуги; послуги із страхування; послуги з маркетингу і надання відомостей щодо продуктів і порядку їх придбання; послуги з пошуку товарів, що найбільше відповідають потребам клієнтів, в тому числі за ціновими характеристиками.

Таблиця 5.2.1.

5.2.2 Аналіз зацікавлених сторін проекту

Група зацікавлених осіб	Інтереси групи в проекті	Умови довгого співробітництва з проектом
Дата-центри обробки трафіку	Забезпечення стабільного отримання трафіку Інтернету в районах «білих плям»	Підтримка та імплементування з legasy кодом попередніх проектів
Підприємства з автоматизації логістики та телекому	Отримання логістичних даних про gps положення користувача для побудови відклику станції	Реалізація автоматизаціх розподілення мережевих обмінів пакетами на основі обхідних мереж
Логічтичні компанії	Отримання цілісності інформації щодо положення користувача.	Реалізація синхронізації з сервісами отримання PUT даних про логістичні характеристики відслідковуваного об’єкту
Наукові дослідницькі центри / заклади	Розвиток та дослідження, покращення даного способу задля наукових та комерційних інтересів	Вміст в роботі теми та тематики мало досліджуваної попередньо, наукова новизна, наявність документації відповідно вимогами, констультування з розробниками
Заклади з великим оборотом інформації	Підтримка реалізації COMMET з'єднань на основі децентралізованих мереж	Підтримка розподілення запитів за асинхронною схемою, обробка запитів на обробку інформації в окремих процесах
Сервіси трекінгу руху транспорту	Покращення якості інформації	Підтримка обходу мереж передачі інформації черех аналогові мережі
Держава	Покращення якості умов життя та функціонування бізнесу	Наявність інвесторів та перспективність проекту
Водії	Отримання інформації від операторів логістичних центрів	Зручний інтерфейс, допомога в роботі

5.2.3. Опис наукового проекту та технології

Науковим продуктом у рамках магістерської дисертації є спосіб знаходження альтернативних джерел сигналу в умовах низької якості мережі.

Таблиця 5.2.2.

5.2.2 Зацікавлені сторони проекту. Загальна оцінка

Група зацікавлених осіб	Важливість	Зацікавленість	Оцінка	Сумарно
Дата-центри	8	7	Висока	56
Підприємства з автоматизації логістики та телекомунікацій	9	4	Низька	36
Логістичні компанії	5	6	Низька	30
Науково-дослідницькі центри / заклади	4	4	Дуже низька	16
Заклади з великим оборотом інформації	3	4	Дуже низька	12
Сервіси трекінгу руху транспорту	3	8		24
Держава	3	3	Дуже низька	9
Водії	7	3	Дуже низька	21
Фізичні особи користувачі	8	8	Висока	64
Інвестори	10	4	Низька	40

Даний спосіб призначений для підтримки доступу до мережі користувачів і базується на самобалансовних графах мережі, що є різновидом децентралізованих бездротових мереж. Ці мережі є самоорганізованими тому, що вони не вимагають заздалегідь створеної інфраструктури, наприклад, як безпровідні мережі або точки доступу

потребують наявності налаштованих маршрутизаторів в керованих безпровідних мережах. Замість цього, кожен вузол ad hoc мережі бере участь в маршрутизації, пересилаючи дані через інші вузли. Таким чином, визначення того, які вузли здійснюють передачу даних, відбувається динамічно на основі архітектури з'єднань мережі.

Самоорганізовані мережі можуть використовувати потоковий алгоритм (Flooding) для передачі даних. В ad hoc мережах усі пристрої мають однаковий статус в мережі і можуть вільно спілкуватися з будь-яким іншим пристроєм цієї мережі в зоні своєї видимості.

Ациклічний направлений граф є випадком орієнтованого графа, в якому відсутні орієнтовані цикли, тобто шляхи, що починаються і закінчуються в одній і тій самій вершині. Орієнтований ациклічний граф є узагальненням дерева.

На ранніх етапах розробки тестування запропонованого методу проводиться методом збору статистичних даних відносно швидкодії за допомого симулятору роботи дата-центру з поточною обробкою даних, наприклад конвертацією чи кодуванням. Даний симулятор буде створюється власноруч засобами написання програмного коду. В наступних етапах застосунку методу дані підприємства будуть слугувати платформою для тестування та збору інформації для подальшого вдосконалення методу, таким чином, розвиток методу ніколи не припиниться.

Для реалізації методу на абстрактному рівні виділено декілька стадій:

1. аналіз різних підходів для прискорення роботи динамічних систем та виділення їх переваг та недоліків;
2. створення систем на основі графу;
3. реалізація створення та самоорганізації мереж на довірених вузлах;
4. реалізація криптографічного алгоритму з відкритим ключем RSA для зв'язку між абонентами а основі шифрування простих чисел;
5. тестування пошуку в ширину в ациклічному графі;
6. реалізація тестування та трастових апаратів в умовах симуляції.

5.2.4.1Резюме продукту

Результатом науково-дослідницької роботи буде ПЗ як набір бібліотек та документація до нього. Дане ПЗ забезпечує організацію однорангових бездротових мереж за допомогою набору бібліотек, що надають API для роботи з сокетами та звертаються до апаратних можливостей платформи в організаціях бездротового зв’язку, забезпечують шифрування на основі відкритого ключа та передачу сигналу за допомогою альтернативних (доступних мереж). При цьому зв’язок з адресатом підтримується на основі попередніх з’єднань. На відміну від пошуку по дереву, пошук по графу може потребувати проходження деяких вершин більше ніж один раз, оскільки не обов'язково є відомості, при переході на вершину, що вона вже була перевірена. З тим як граф стає більш щільним, ця надмірність стає проявлятись більше, що призводить до збільшення необхідного часу на обчислення. Таким чином, краще запам'ятовувати, які вершини було вже пройдено алгоритмом, так щоб вершини перевірялися не часто, якщо це можливо (таке запам'ятовування дозволяє уникнути найгіршої поведінки — нескінченного пошуку по циклу). Це можна зробити за допомогою асоціації з кожною вершиною «колір» або стан «проходження» вершини під час пошуку, який потім помічається і оновлюється алгоритмом при проходженні кожної вершини графу. Бібліотека та програмне забезпечення може швидко будувати та масштабувати рішення на основі графів ациклічного типу та реалізувати програмні прийоми пошуку адресата по такій мережі. Також далі будуть розглянуті можливості та шляхи вводу ПЗ на виробництва різних типів, а також можливості отримання ПЗ та подальшого використання у «домашніх умовах», також будуть описані необхідні вимоги для встановлення ПЗ.

5.2.4.2 Опис продукту

Результатом дослідницької роботи буде ПЗ, але слід зазначити, що ПЗ далеко не основний виробничий продукт, в основі ПЗ лежить спосіб перемикання та відправлення відповіді на запам’ятований порт в мережі через довгі з’єднання, тобто результатом роботи буде саме спосіб, який можна встановлювати або інтегрувати в різні системи виробництва, що працюють з телекомунікаційними чи логістичними рішеннями.

Заклади з великим обігом інформації, використовуючи дані технології, зможуть в рази швидше сортувати зберігати та перенаправляти необхідну інформацію по мережах у випадку критичного перенавантаження мережі в пікові години або під час захисту від атак різного типу.

Основною особливістю розробки є підхід не в збільшенні потужності сигналу або збільшенні чутливості, а програмне алгоритмічне рішення, що реалізує імітацію основного ходу трафіку за допомогою децентралізованих однорангових вузлів.

Продукт йтиме разом з веб-версією, що працює в тій самій мережі що і всі пристрої, сама мережа необхідна для організації, зв’язок та налаштувань вузлів, що будуть анонімно передавати інформацію в мережі між частинами мережі та необхідні для балансування та налаштовування мережі.

Продукт знаходиться на стадії постійного розвитку, що дасть можливість постійного доповнення методу та пристосування його до різних типів даних, що є однією з проблем оптимізації роботи кластерних систем, адже різні типи інформації та різні системи виконують обробку та зберігають по-різному. Бізнес-рішенням є постійна підтримка та постійний розвиток саме методу, адже ПЗ буде встановлюватись саме на момент дійсної версії методу, тому за оновлення та підтримку необхідна додаткова фінансова підтримки та безпосередньо за отримання нової версії продукту. Таким чином, оскільки результатом роботи окрім безпосереднього прискорення є статистичні дані кожного власника системи, система обробки

інформації буде використовувати власний спосіб зберігання і прискорення, тому затримок у оновленні даних буде менше, що дасть можливість своєчасно та оперативно помічати будь-які зміни у системах, що обслуговуються.

В майбутньому спосіб перейде на інтегрування на великі телекомунікаційні частини, що значно прискорить обмін трафіку, але основним розвитком є перетворення на платформу для запуску додатків, що будуть працювати стабільніше в мережі з низьким пороговим сигналом.

5.2.5 Ціннісна пропозиція продукту

Цінність – це розуміння користі, яку покупець отримує від продукту. З такої позиції можна чітко охарактеризувати цінність продукту для кожної зацікавленої сторони, або для кожного потенційного користувача, агрегувти ці відомості та побудувати детальну інформацію.

Так як цінність є суб’єктивною величиною, то далі неведена цінність відносно зацікавлених осіб.

- Дата центри обробки трафіку
  - Забезпечити стабільне отримання трафіку інтернету в районах «білих плям».
  - Розпаралелювання виконуваних обмінів інформацією через з’єднання, а саме автоматичне розпаралелювання конвертування коду в багато-потоковий, що працює на паралельному комп'ютері, наприклад, на SMP або NUMA машині та перенесений на передатчик зв’язку. Повне розпаралелювання послідовних з’єднань залишається занадто складним завданням, що вимагає найскладніших видів аналізу програм.
- Підприємства з автоматизації логістики та телекомунікацій

- Отримання логістичних даних про gps положення користувача для побудови відгуку станції. Для передачі даних положення використовується стільниковий зв'язок GSM та такі його сервіси як GPRS, EDGE, SMS або CSD. Програма дає можливість постійно підтримувати з’єднання та спостерігати за рухом об'єкту. Може бути використана для визначення розташування: автотранспорту (автомобільні трекери, що під'єднуються до бортової системи автомобіля), людей (персональні трекери), тварин (надкомпактні, дуже легкі, кріпляться безпосередньо до нашийника). Інформація про місцезнаходження може передаватися в реальному часі, через певні інтервали або за запитом.

- Логістичні компанії
  - Отримання цілісності інформації щодо положення користувача.
  - Покращення контролю за пресуванням транспорту.
  - Покращення спостереження за дітьми чи людьми похилого віку.
  - Покращення спостереження за підлеглими. GPS-контроль забезпечує постійну фіксацію маршрутів пересування робітників, наприклад, торгових представників, водіїв, мерчендайзерів тощо. Таким чином, можна слідкувати, чи дотримується персонал компанії заданого маршруту. Це все допомагає оптимізувати робочий процес, знизити нецільове використання робочого часу.
  - Реалізує підтримку в місцях зі слабким сигналом «SOS», яка дозволяє відправити сигнал тривоги у вигляді SMS-повідомлення з точними координатами.
  - Підтримка long-pooling з'єднань
- Наукові дослідницькі центри / заклади
  - Розвиток та дослідження, покращення даного способу задля наукових та комерційних інтересів.

- Заклади з великим оборотом інформації
  - Підтримка реалізації COMMET з'єднань на основі децентралізованих мереж, а саме Long-polling, коли клієнт підключається до сервера, який не закриває з'єднання, доки не з'являться дані або мине час очікування. Після чого клієнт підключається повторно. Streaming: в цьому випадку з'єднання постійно залишається відкритим і не закривається після кожної передачі даних. Цей підхід є складнішим і потребує спеціально програмного забезпечення. Реалізувати таку модель на стороні клієнта можна з допомогою JavaScript використовуючи AJAX або IFRAME. А на стороні сервера з допомогою, практично, будь-якого веб-сервера та мови програмування.
- Сервіси трекінгу руху транспорту
  - Покращення якості інформації за допомогою підтримки постійного зв’язку іншими точками зв’язку.
- Водії
  - Отримання інформації від операторів логістичних центрів
- Фізичні особи користувачі
  - Покращення User Experience та забезпечення цілісності інформаційного відклику у автономних пристроях.

5.2.6 Прибутки та витрати бізнес продукту

Дана розробка має декілька потенційних джерел доходів, які наведені в таблиці 5.6.1.

Мікротранзакції – це популярна бізнес-модель розповсюдження завантажуваного контенту або доступу до надаваних послуг за невеликими цінами (зазвичай в межах 10 $). Така модель особливо поширена в різних MMORPG-іграх і часто замінює собою плату у вигляді підписки. Також подібна модель оплати застосовується при поширенні контент-послуг в мережах стільникового зв'язку (мелодії, картинки, програми для мобільних

Таблиця 5.6.1

Таблиця доходів та витрат продукту

Доходи	Умовні одиниці
Мікротранзакції	20
Організація трафіку	4
Контекстний аналіз на основі трафіку	6
Контекстний аналіз на основі місцеположення користувача	10
Контекстний аналіз на основі контактів	4
Автоматизація телекомунікацій комнацій на основі бездротових рішень	100
Імплементування рішень в сферу логістики щодо покарщення якості зв'язку та підтримки постійних long-puuling з'єднань	500
Монетиація на основі продажу Ліцензій на використання програмних рішень Бібліотеки для розробки стороніх програм	80
Монетизація на основі рішень впровадження організації децентралізованих анонімних мереж для компаній що використовують логістичні дані для конкуренції на ринку	500
Налаштовування мереж	10
Продажа ліцензійних ключів для налаштовування продукту(бібліотеки)	20
Розробка програмного мультиплатформеного продукту	-100
Збереження даних в датацентрах	-200
Маркетинг	-400
Закупка ліцензії на доступ до програмного коду мобільних платформ	-20
Закупівля криптостійкого трафіку	-20
Бібліотеки шифрування	-10
Автоматична асинхронна модель (розробка(покращення))	-40
Закупівля апаратних рішень для тестування	-100
Інші витрати	-100

телефонів і комунікаторів). Невисокий розмір оплати знижує психологічний бар'єр перед покупкою у потенційного покупця.

1. Контекстний аналіз на основі трафіку виявлення і пошук, визначення властивостей та характеристик на основі зібраних статистичних даних

або емпіричних досліджень окремих об'єктів або явищ. Використовуються з метою встановлення логічних закономірностей які впливають на досліджувані об'єкти або явища, і пошуку переваг та вразливостей які можуть виявлятися під впливом факторів.

2. Контекстний аналіз на основі місцеположення користувача.
3. Автоматизація телекомунікацій на основі бездротових рішень
4. Імплементування рішень в сферу логістики щодо покращення якості зв'язку та підтримки постійних long-puuling з'єднань. Монетизація на основі рішень впровадження організації децентралізованих анонімних мереж для компаній що використовують логістичні дані для конкуренції на ринку.

На малюнку 5.6.1 наведено графік фінансового розвитку продукту за умови початкових, стартових інвестицій в нього.

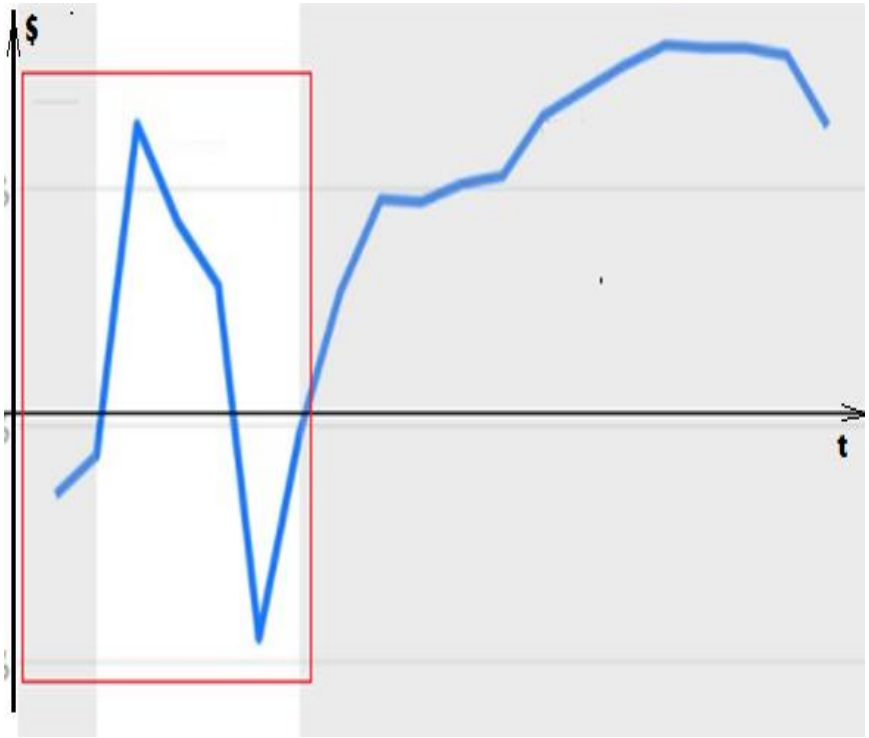


Рис. 5.6.1. Графік доходів

На графіку зображено та виділено червоним квадратом зону існування проекту на базі початкових, стартових інвестицій в нього. Доходи в основному йдуть за рахунок постійних мікротранзакцій трафіку

користувачів та контекстного аналізу, різкі стрибки доходів – це отримання замовлення, встановлення інмплементаційного програмного забезпечення для компаній. Стрибки вниз– це помісячна оплата за послуги. Впродовж постійних ітерацій прибуток постійно збільшується та при збільшенні клієнтської бази та децентралізованої мережі, збільшується і початковий дохід від мікротранзакцій та обслуговування мереж.

Реальний прибуток складає 15% від доходів в песимістичному сценарії, що можна вважати хорошим результатом.

Показники можна збільшити за рахунок акцій та виставок з участю даного продукту.

5.2.7 Бізнес модель

Бізнес-модель – це те, як ми організовуємо нашу діяльність, щоб надати замовнику обіцяну цінність. Це те, як ми організовуємо наші внутрішні операції, щоб виробляти і доставляти цінність замовнику. Нашими клієнтами можуть бути або кінцеві споживачі (якщо у ланцюжку створення цінності ми знаходимося ближче до кінцевої ланки, клієнта), або інші організації або суб’єкти ланцюжка залежно від того, де ми знаходимося у ньому.

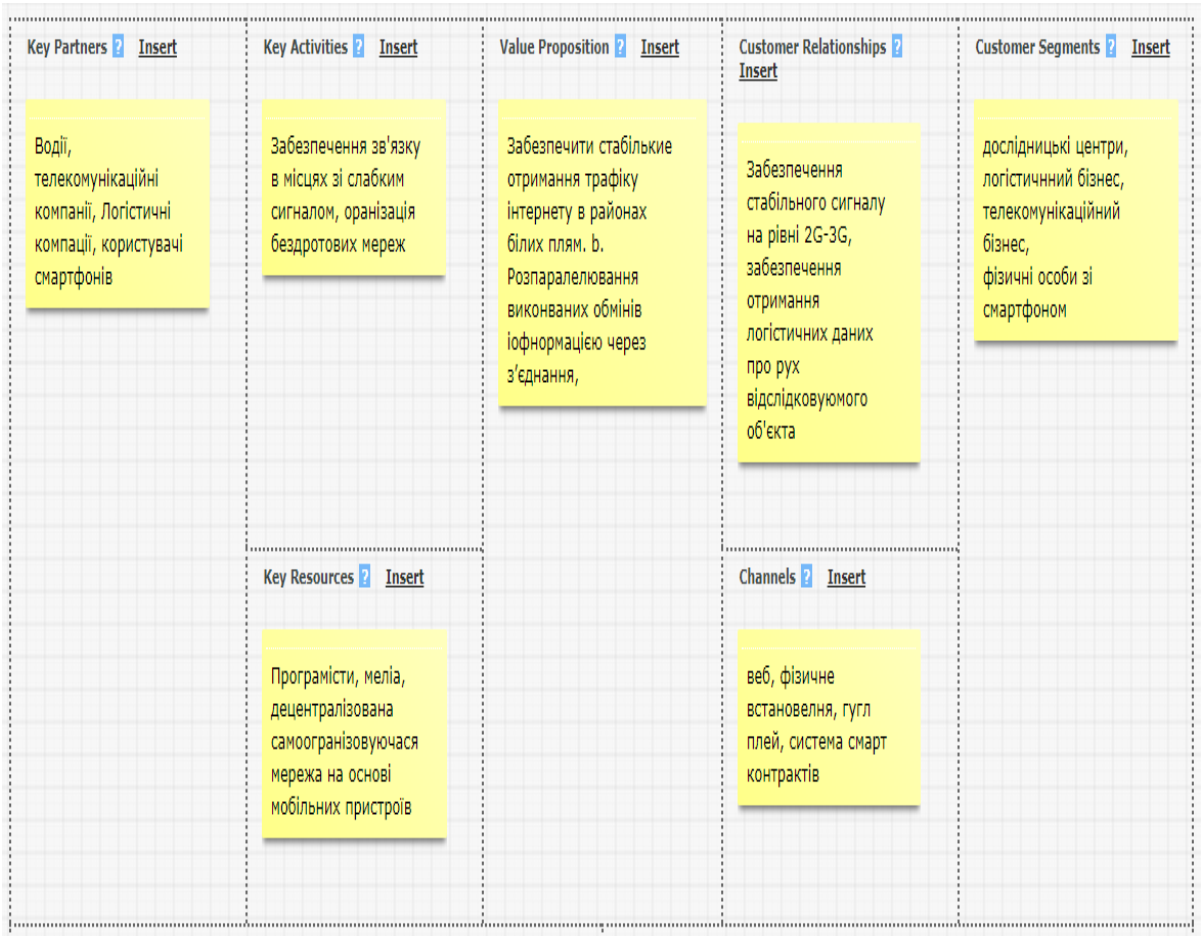


Рис. 5.7.1. Канва бізнес-плану 1

А саме це те, як ми організовуємо діяльність для реалізації цієї цінності, та які є грошові елементи є у цій цінності, щоб створити модель ціноутворення і, відповідно, встановити ціну на наші послуги та продукти.

1. Ключовими партнерами є зацікавлені сторони.
2. Ключовою діяльністю є забезпечення зв’язку в місцях зі слабким сигналом мережі.
3. Ціннісна пропозиція є забезпечення стабільного отримання трафіку інтернету в районах «білих плям». Розпаралелювання виконуваних обмінів конформацією через з’єднання, а саме автоматичне розпаралелювання конвертування робочого коду або запитів у рамках динамічно побудованого графу користувачів та підтримувати їх вимоги як цілісну сутність, надаючи їй всі необхідні елементи та ресурси для існування.

4. Клієнтським сегментом є дослідницькі центри, логістичні бізнеси, телекомунікаційні бізнеси, фізичні особи.
5. Ключовими ресурсами є програмісти, медіа, децентралізована самоорганізовуюча мережа на основі мобільних пристроїв.
6. Канали збуту, веб, фізичне встановлення, Google Play, система смарт контрактів.
7. Ціннісна структура – це децентралізована мережа, набір функціональностей, що підтримують її та загальні положення про виконання протоколів COMMET.

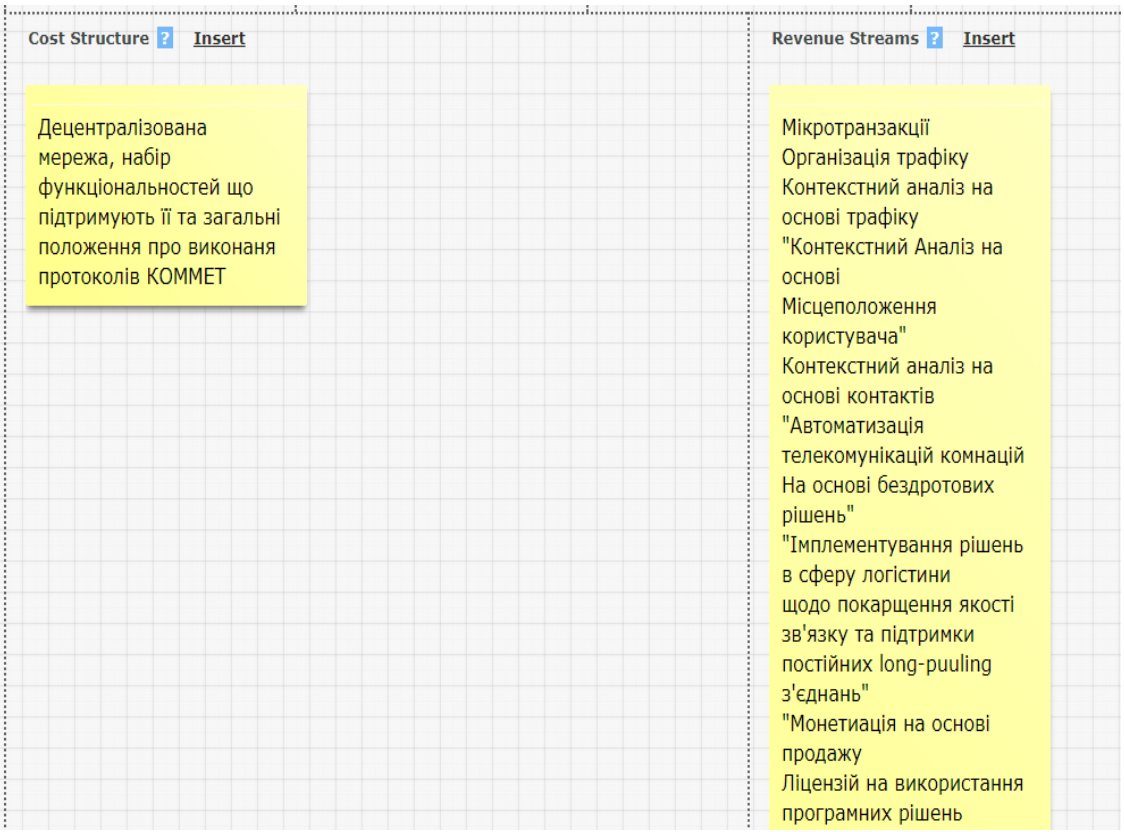


Рис. 5.7.2. Канва бізнес-плану 2

5.3 Висновки

На основі проведеного дослідження можливо побудувати бізнес-схему з прибутками на територіях країн, що розвиваються, та для людей, що мають проблеми із якістю зв’язку.

Основною цінністю даного продукту є метод знаходження альтернативного сигналу в умовах відсутності зв’язку та в умовах знаходження абонента в зоні «білих плям».

Розроблений програмний метод має свої джерела прибутку та канали збуту продуктів, також в основу способу покладено роботу з децентралізованою мережею однорангових вузлів.

Метод легко маштабується як у вигляді бібліотеки, так і як веб застосунок для налаштовування мереж для забезпечення виконаня вимог клієнтів. На даний момент складається патентна документація, що пришвидшить імплементування даної розробки в продуктивне середовище.

ВИСНОВКИ

У даній магістерській роботі виконані наступні завдання.

1. Проведено аналіз існуючих методів безперервної передачі даних в умовах швидкого розгортання бездротової мережі для пристроїв з ОС Android а також наявних комерційних продуктів, що використовують алгоритми роботи з одноранговими мережами.

2. Визначено основні характеристики, переваги та недоліки існуючих методів безперервної передачі даних в умовах швидкого розгортання бездротової мережі, на основі даних результатів визначено можливі шляхи модифікації методів.

3. Обрано метод для модифікації та досліджено можливі шляхи його удосконалення.

4. Сформульовано модифікований метод безперервної передачі даних в умовах швидкого розгортання бездротової мережі такий, що підтримує ряд покращень його характеристик.

5. Реалізовано програмне забезпечення, яке імплементує в собі даний метод.

6. Проведено порівняльний аналіз ефективності сформульованого методу в порівнянні з існуючими аналогами та виконано оцінку отриманих результатів.

7. Сформульовано шляхи подальшого удосконалення запропонованого методу.

Результати, отримані при реалізації методу, показали, що запропонований метод дає кращі результати в зазначених характеристиках, а саме час затримки було зменшено, причому використання додаткової пам’яті для реалізації покращень є незначною.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Guowang Miao, Jens Zander, Ki Won Sung, and Ben Slimane, Fundamentals of Mobile Data Networks, Cambridge University Press, ISBN 1107143217, 2016.
2. Perkins, C.E. Ad hoc On-demand Distance Vector (AODV) Routing / C.E. Perkins, C.E. Beldong-Royer // RFC 3561. – July 2003.
3. Broch, J.A performance comparation of multihop wireless ad hoc network routing protocols / J. Brocj, D.A. Maltz // Proc. Of MOBICOM`98 -1998.
4. Basagni, S. Mobility – Adaptive Protocols for Managing Large Ad Hoc Networks / S. Basagni. D. Turgut // Proceedings of the IEEE International Conference on Communication (ICC) – 2001 – p. 1539-1543.
5. Kawadia, V. System services for implementation Ad-Hoc routing: Architecture. Implementation and Experiences / V. Kawadia, Y Zhang, B, Gupta // Proceeding of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys) San Drancisko, C.A. – June 2010 – P. 99-112.
6. RFC 7541 — HPACK: Header Compression. Режим доступу https://tools.ietf.org/html/rfc7541
7. An Extensive Examination of Data Structures. Режим доступу: https://msdn.microsoft.com/en-us/library/ms379574(v=vs.80).aspx, вільний
8. Modeling mobility for vehicular ad-hoc networks Режим доступу: https://dl.acm.org/citation.cfm?id=1023892
9. Akyildiz I. F., Wang X., Wang W. Wireless mesh networks: a survey //Computer networks. – 2005. – Т. 47. – №. 4. – С. 445-487.
10. Perkins C. E. et al. Performance comparison of two on-demand routing protocols for ad hoc networks //IEEE Personal communications. – 2001. – Т. 8. – №. 1. – С. 16-28.

11. Ko Y. B., Vaidya N. H. Location-Aided Routing (LAR) in mobile ad hoc networks //Wireless networks. – 2000. – Т. 6. – №. 4. – С. 307-321.
12. Ko Y. B., Vaidya N. H. Location-Aided Routing (LAR) in mobile ad hoc networks //Wireless networks. – 2000. – Т. 6. – №. 4. – С. 307-321.
13. Daly E. M., Haahr M. Social network analysis for routing in disconnected delay-tolerant manets //Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing. – ACM, 2007. – С. 32-40.
14. Karp B., Kung H. T. GPSR: Greedy perimeter stateless routing for wireless networks //Proceedings of the 6th annual international conference on Mobile computing and networking. – ACM, 2000. – С. 243-254.
15. Intanagonwiwat C., Govindan R., Estrin D. Directed diffusion: A scalable and robust communication paradigm for sensor networks //Proceedings of the 6th annual international conference on Mobile computing and networking. – ACM, 2000. – С. 56-67.
16. Trifunovic S. et al. WiFi-Opp: ad-hoc-less opportunistic networking //Proceedings of the 6th ACM workshop on Challenged networks. – ACM, 2011. – С. 37-42.
17. Conti M. et al. Experimenting opportunistic networks with WiFi Direct //Wireless Days (WD), 2013 IFIP. – IEEE, 2013. – С. 1-6.
18. Arnaboldi V., Conti M., Delmastro F. CAMEO: A novel context-aware middleware for opportunistic mobile social networks //Pervasive and Mobile Computing. – 2014. – Т. 11. – С. 148-167.
19. Casetti C. et al. Content-centric routing in Wi-Fi direct multi-group networks //World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a. – IEEE, 2015. – С. 1-9.
20. Broch J. et al. A performance comparison of multi-hop wireless ad hoc network routing protocols //Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking. – ACM, 1998. – С. 85-97.

21. Perkins C., Belding-Royer E., Das S. Ad hoc on-demand distance vector (AODV) routing. – 2003. – №. RFC 3561.
22. Lee S. J., Gerla M. AODV-BR: Backup routing in ad hoc networks //Wireless Communications and Networking Confernce, 2000. WCNC. 2000 IEEE. – IEEE, 2000. – T. 3. – C. 1311-1316.
23. Chakeres I. D., Klein-Berndt L. AODVjr, AODV simplified //ACM SIGMOBILE Mobile Computing and Communications Review. – 2002. – T. 6. – №. 3. – C. 100-101.

ДОДАТКИ

Додаток 1

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

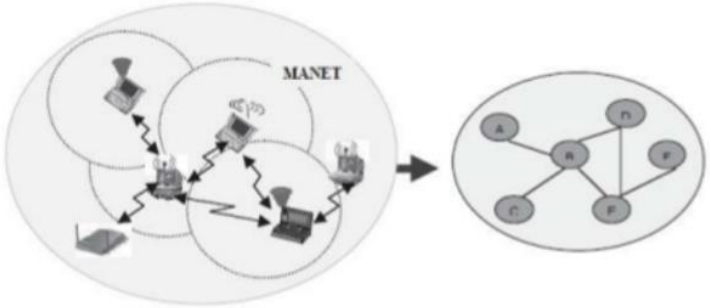
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

МОДИФІКОВАНИЙ МЕТОД БЕЗПЕРЕРВНОЇ ПЕРЕДАЧІ ДАНИХ В УМОВАХ ШВИДКОГО РОЗГОРТАННЯ МЕРЕЖІ ДЛЯ ПРИСТРОЇВ З ОС ANDROID

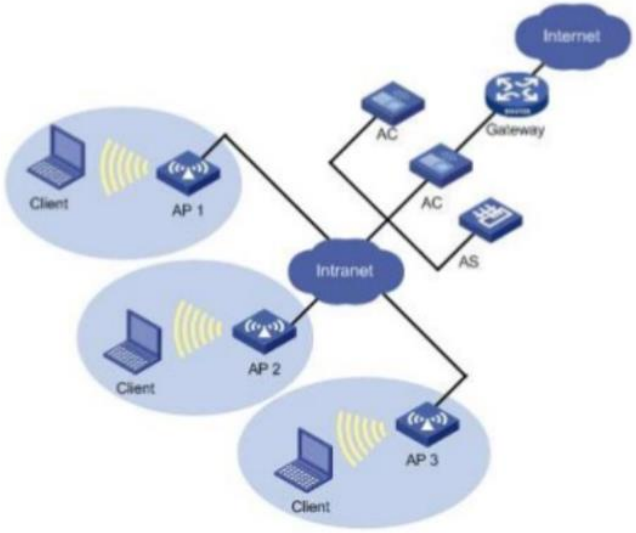
Виконав: Петрусь В. І., КП-71 пм
Науковий керівник: Олещенко Л. М.

Київ – 2018

Актуальність



MANETs



WLAN

Наукове завдання

Удосконалити програмні методи безперервної передачі даних в однорангових бездротових мережах.

Мета дослідження

Оптимізувати існуючі програмні рішення підтримки безперервної передачі даних в умовах швидкого розгортання мережі для пристроїв з OS Android

Об’єкт дослідження:

процес програмної організації та підтримки безперервної передачі даних в умовах швидкого розгортання мереж.

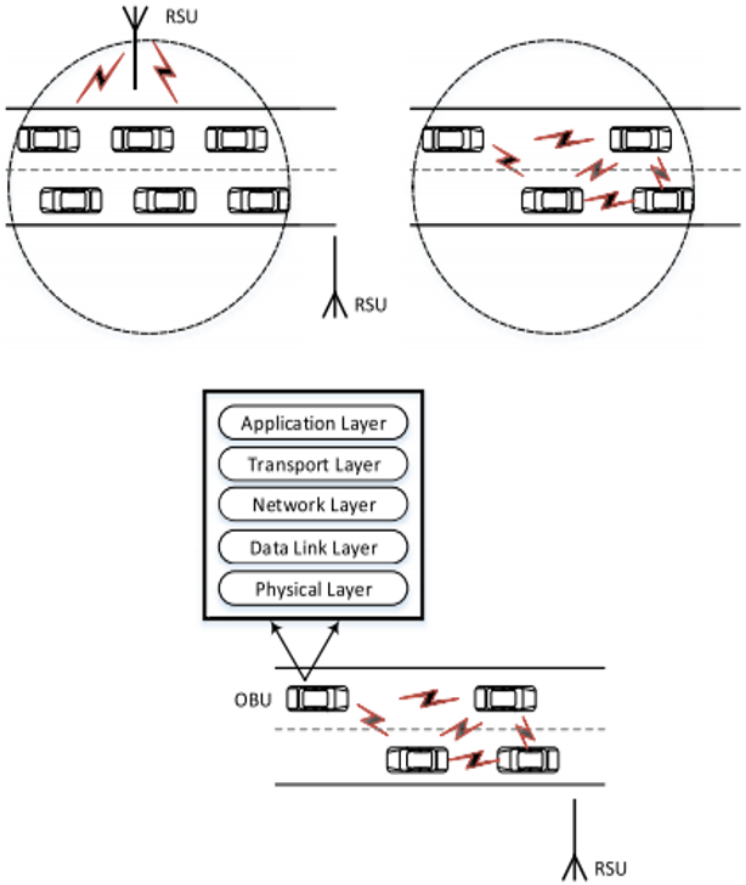
Предмет дослідження:

методи програмної оптимізації маршрутизації та безперервної передачі даних в однорангових бездротових мережах.

Проблематика

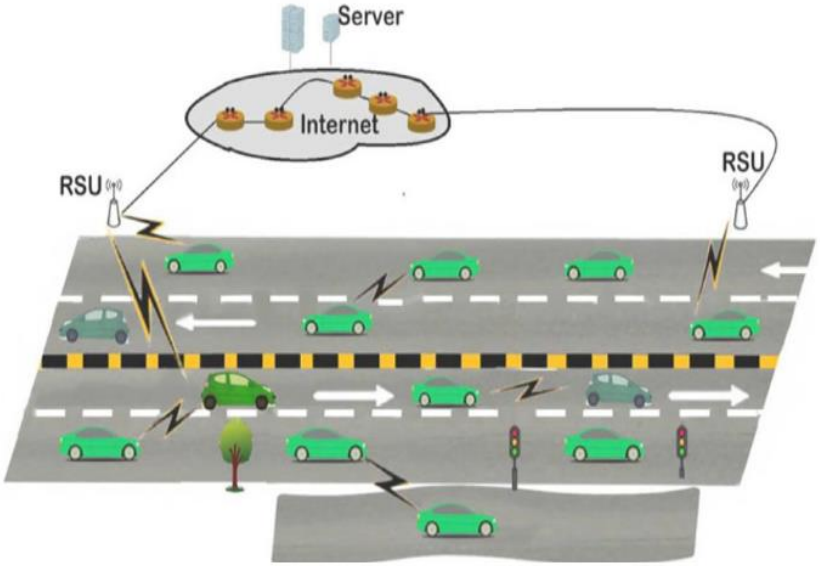
- Програми та сервіси, що використовують бездротові мережі, не завжди є нерухомими, і умови їх взаємодії між собою не завжди однакові. Особливо це наглядно при використанні децентралізованих мереж.

- Більшість алгоритмів розподілення сигналу в мережі не пов’язані з децентралізованими мережами, граф в такій мережі буде динамічно розширюватись та конфігурувати сам себе, що не є тривіальною задачею з оптимальними розв’язками.



Децентралізована система

- учасники поділяють всі необхідні обчислення між собою, не вдаючись до ресурсів централізованих систем;
- дані мережі зберігаються на окремо взятих пристроях самих учасників;
- дуже висока стійкість до зломів даних учасників або атак на всю мережу.



Задачі

1. Аналіз варіантів архітектури організації одноранговых мереж з використанням мобільних пристроїв.
2. Аналіз протоколів динамічної маршрутизації та розробка системи критеріїв оптимізації мережі.
3. Розробка модифікованого методу для оптимізації показників мережі згідно наступних критеріїв та обмежень: кількість вузлів, які займаються маршрутизацією (що вливає на складність мережі); наявність у кожного вузла маршруту до будь-якого іншого вузла в мережі, можливість виникнення петель, рівень знання кожним вузлом топології всієї мережі, кількість ресурсів для знаходження резервного каналу зв’язку.
4. Моделювання бездротової мережі з підтримкою стану втрати мінімальної кількості пакетів під час обміну інформацією.
5. Тестування та порівняння результатів дослідження.
6. Створити стартап на основі імплементації розробленого програмного модифікованого методу.

Проблеми використання ANET

- При використанні мереж на мобільних рухомих пристроях виникає проблема з пошуком резервного каналу зв’язку.

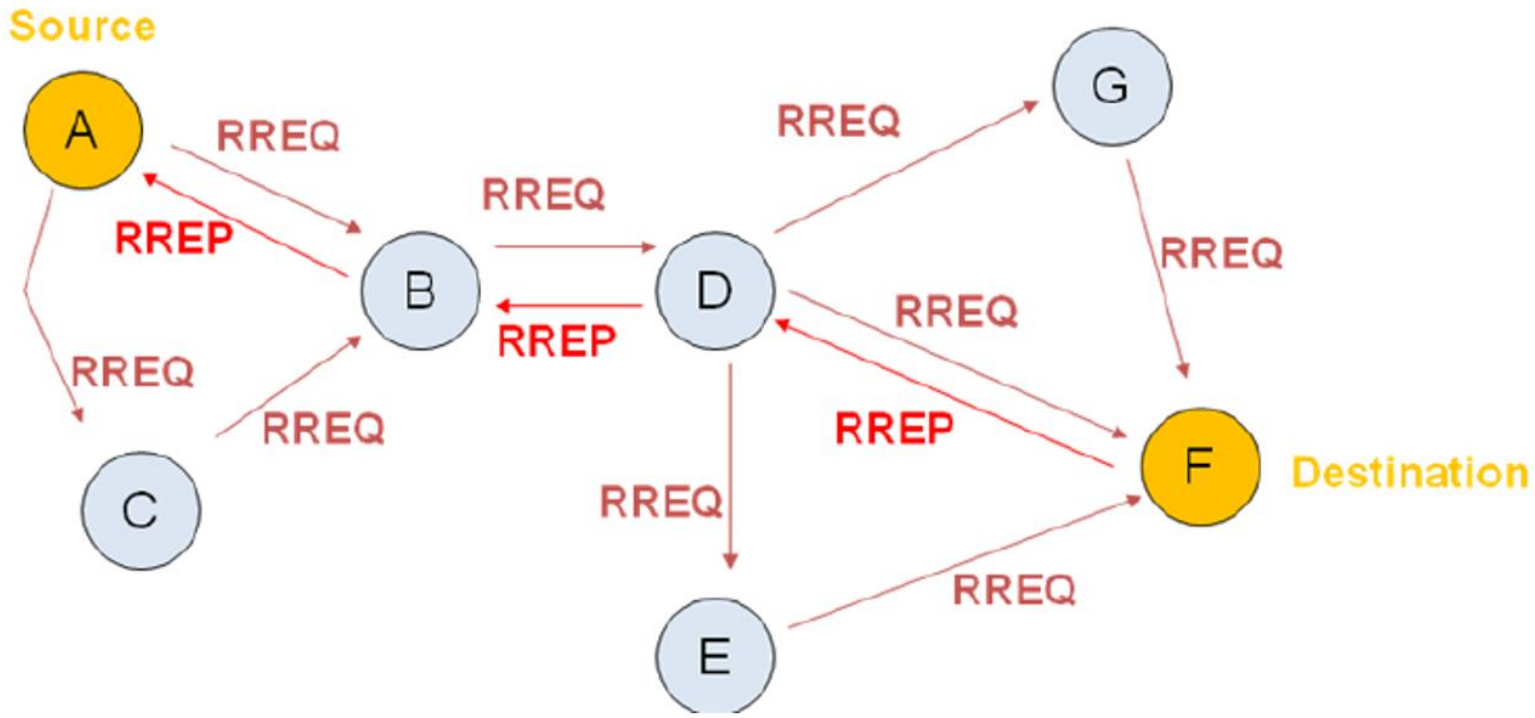
- Постійне балансування мережі використовує значну частину обчислювальних можливостей апаратної частини кожного вузла для підтримання мережі.

- Індексування повідомлень в умовах пошуку резервного каналу та перевантаженості мережі може створювати колізії з передачі пакетів всередині мережі.

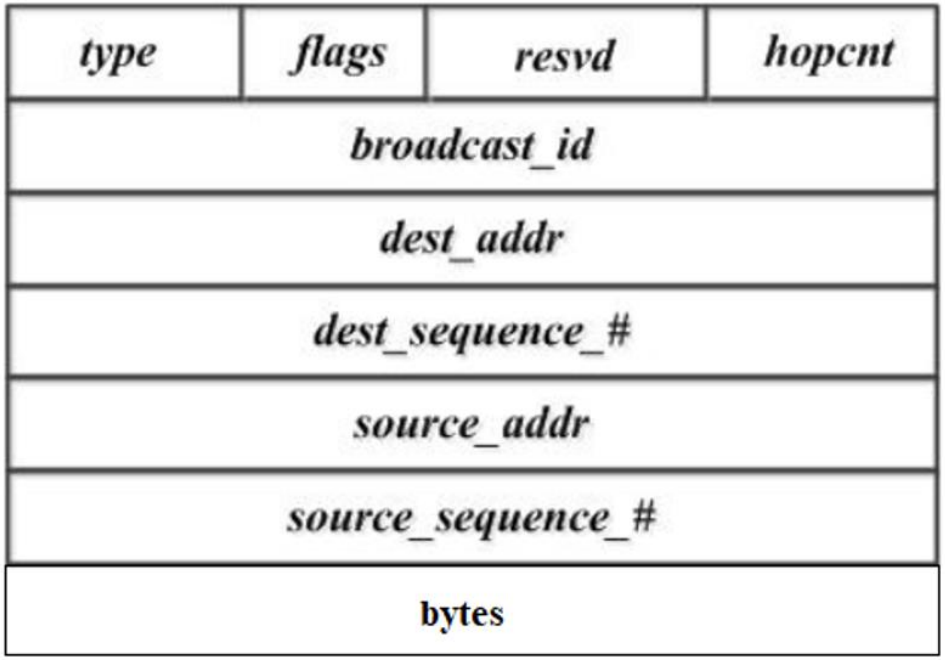
Існуючі методи динамічної маршрутизації

- AODV - *Ad hoc On-Demand Distance Vector*
- OSPF - Open Shortest Path First
- RIP- Routing Information Protocol

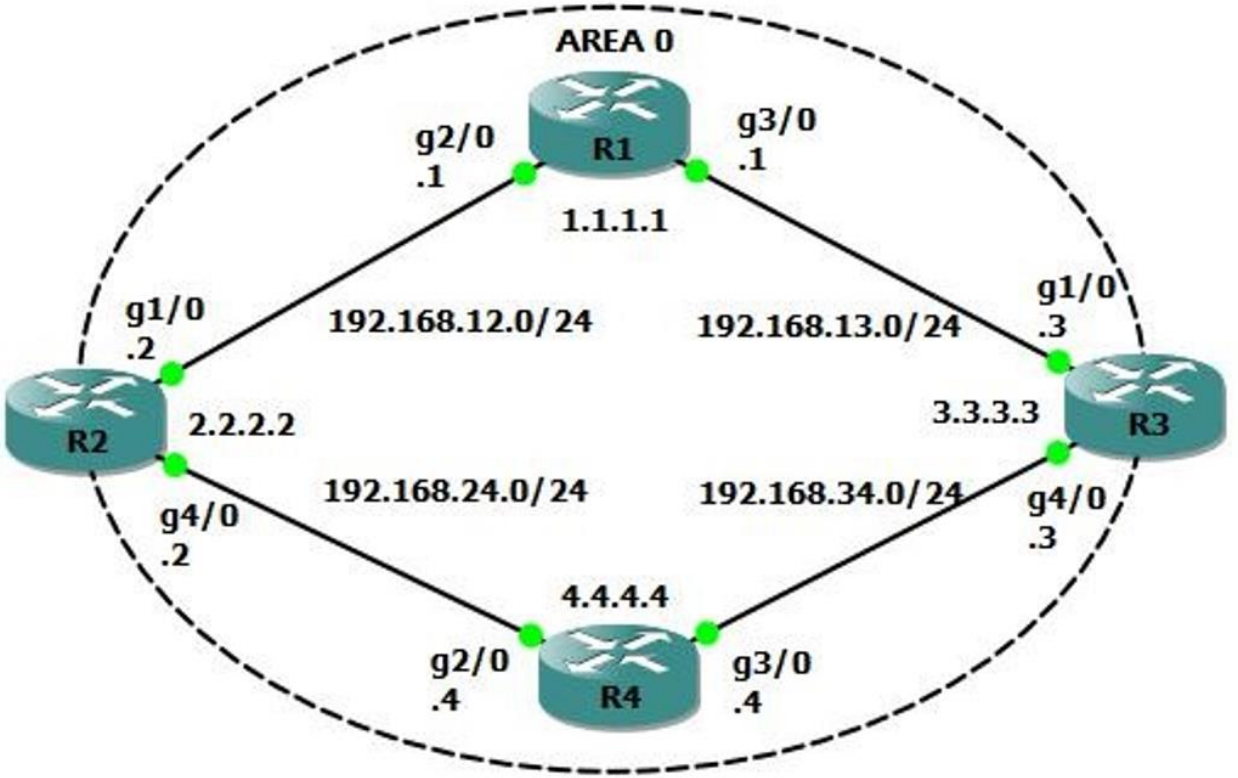
AODV


Source
A
RREQ
RREP
B
RREQ
RREP
D
RREQ
G
RREQ
RREQ
RREQ
RREQ
C
RREQ
RREP
F
Destination
RREQ
RREQ
E

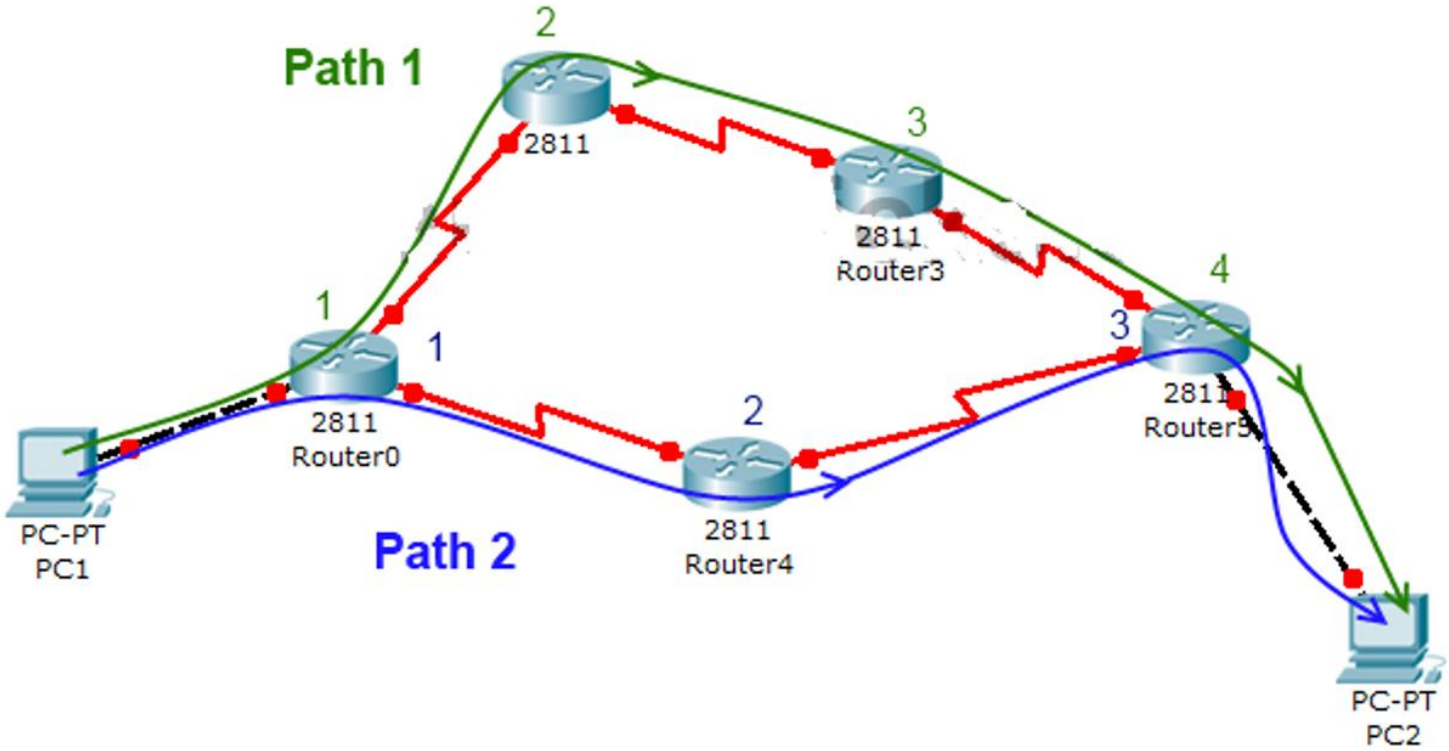
Структура даних повідомлення RREQ



OSPF


AREA 0
R1
g2/0
.1
g3/0
.1
1.1.1.1
g1/0
.2
192.168.12.0/24
192.168.13.0/24
g1/0
.3
R2
2.2.2.2
3.3.3.3
R3
192.168.24.0/24
192.168.34.0/24
g4/0
.2
g4/0
.3
4.4.4.4
g2/0
.4
R4
g3/0
.4

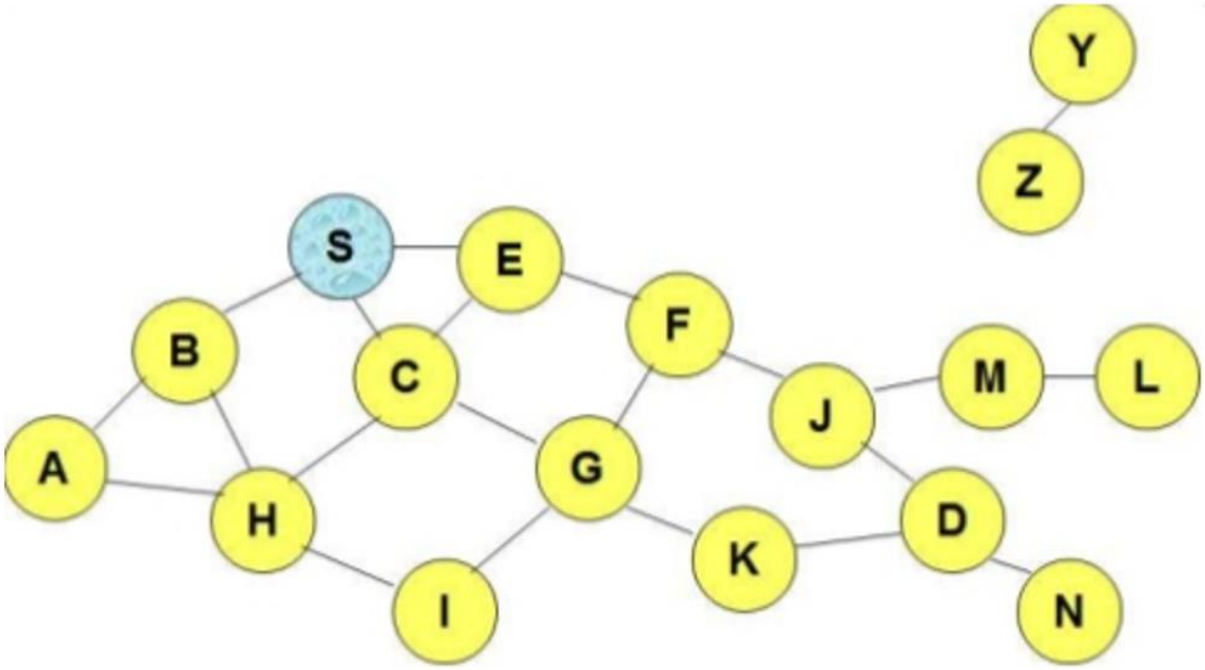
RIP


Path 1
2
2811
3
2811
Router3
4
3
1
1
2811
Router0
2
2811
Router4
Path 2
PC-PT
PC1
PC-PT
PC2

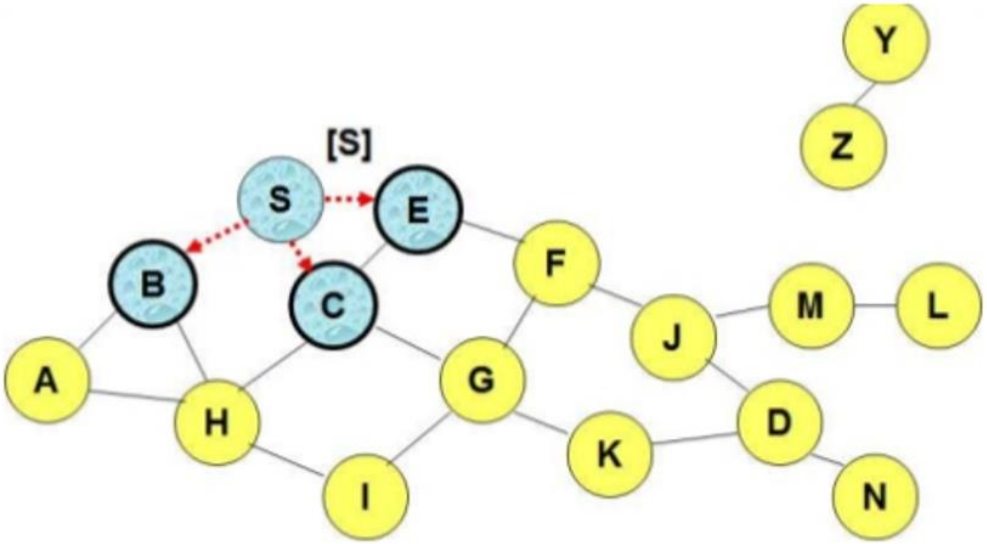
Модифікований програмний метод

- Кожному вузлу при підключенні до мережі автоматично присвоюється ідентифікатор мережі.
- Для кожного вузла обчислюється вартість та шлях до сусіднього вузла, ця інформація зберігається в таблиці маршрутизації:
  - програш в пам’яті,
  - виграш в стабільності мережі та швидкодії.
- Мультиплексування потоків за допомогою бінарного слою фреймів.
- Стискання заголовків передачі даних методом HPACK.
- Додавання проміжної синхронізації вузлів в «густих мережах».
- Застосування модифікованого алгоритму Беллмана-Форда для балансування вузлів з обгорткою для рухомих мереж(рекурсивний перерахунок зворотного шляху).
- Зберігання в кожному вузлі історії руху даних та часткова синхронізація її (n<3), коригування зворотнього шляху на основі проміжних таблиць.

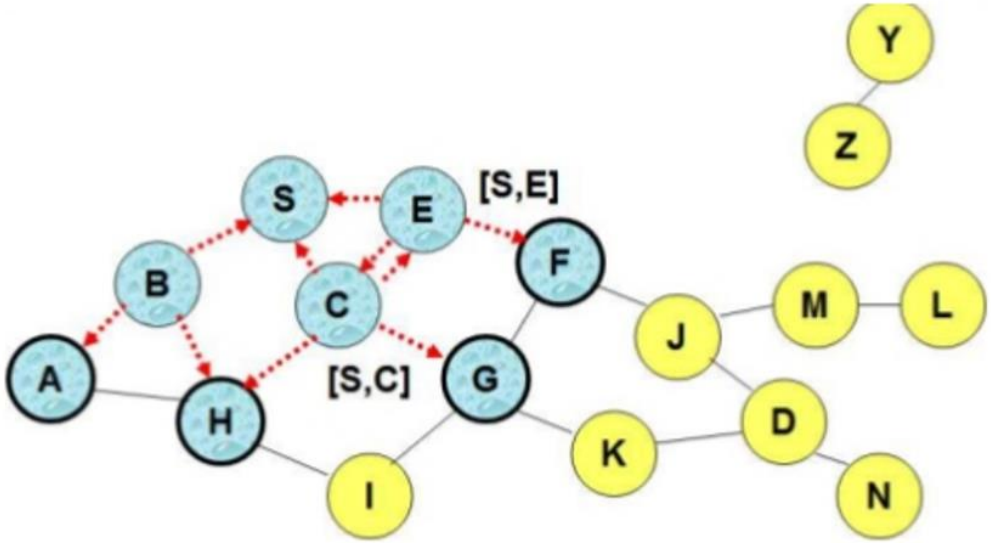
Схема передачі даних



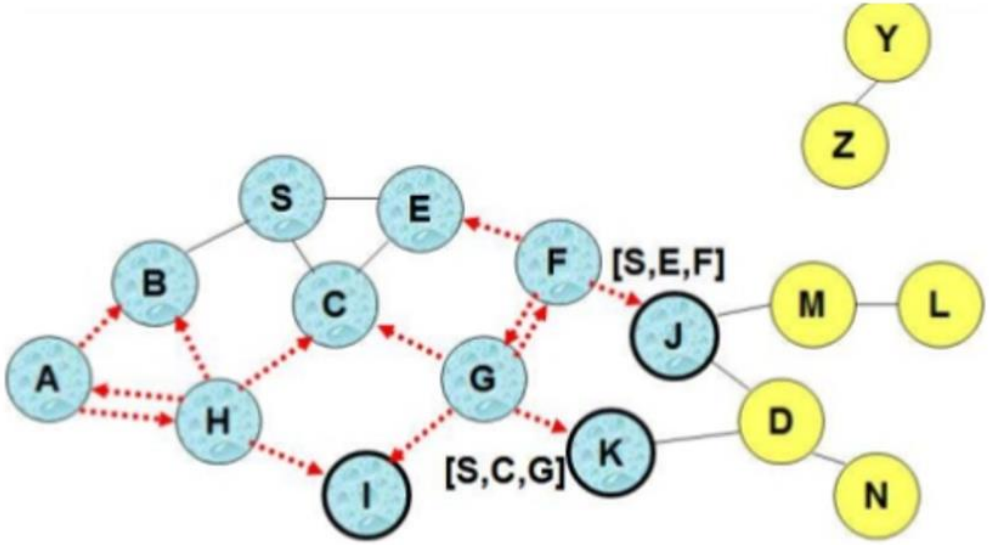
Пошук адресата повідомлення в мережі. Етап 1



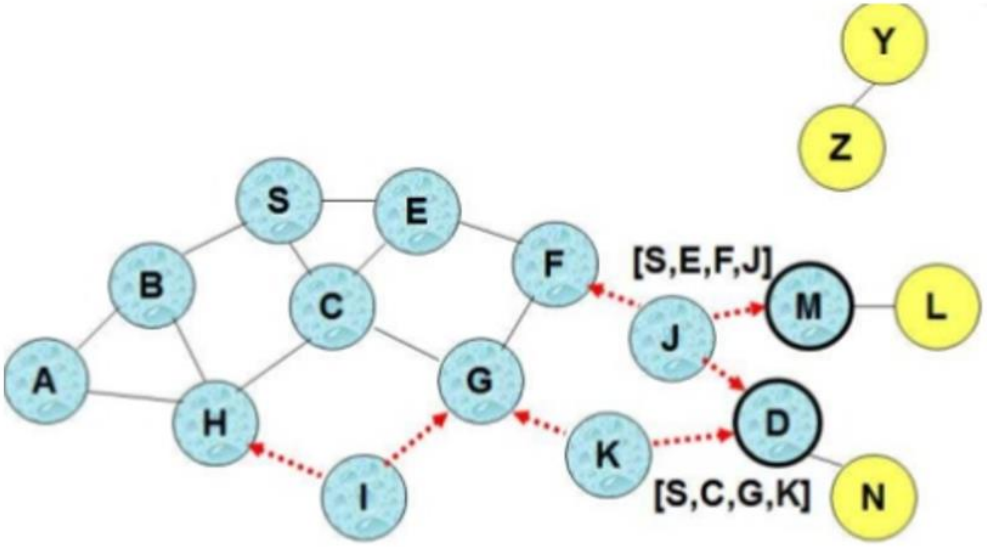
Пошук адресата повідомлення в мережі. Етап 2



Пошук адресата повідомлення в мережі. Етап 3



Пошук адресата повідомлення в мережі. Етап 4



Пошук адресата повідомлення в мережі. Етап 5

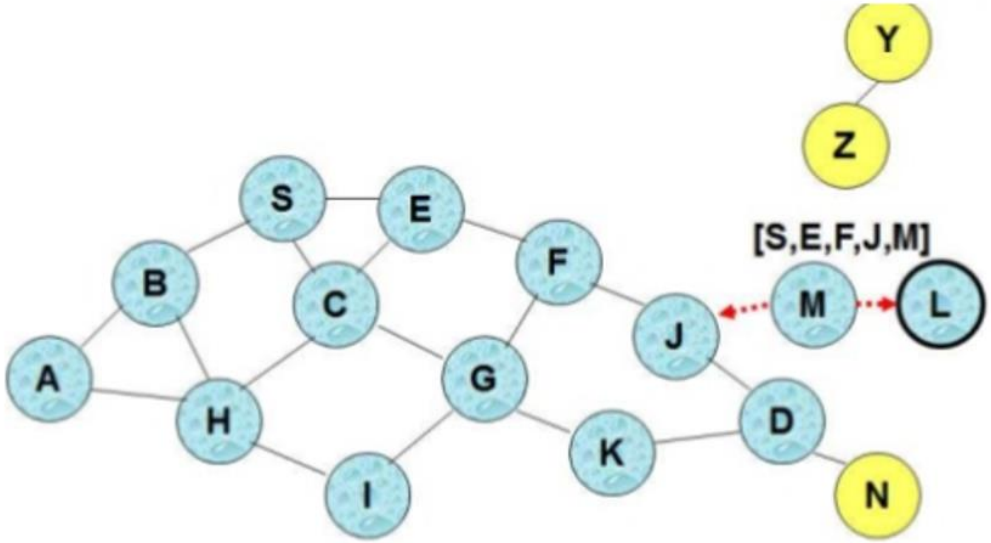
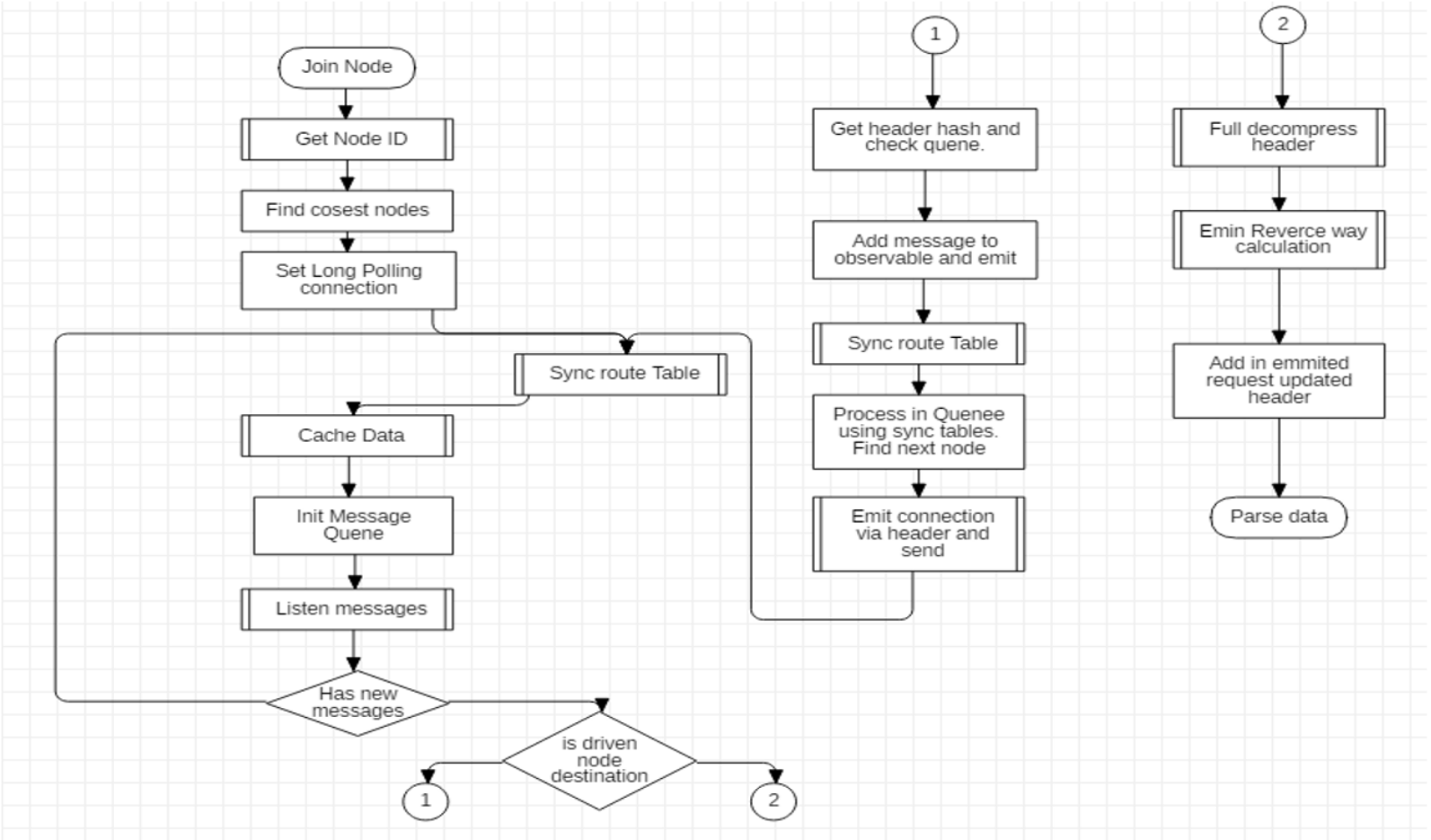


Схема алгоритму модифікованого методу

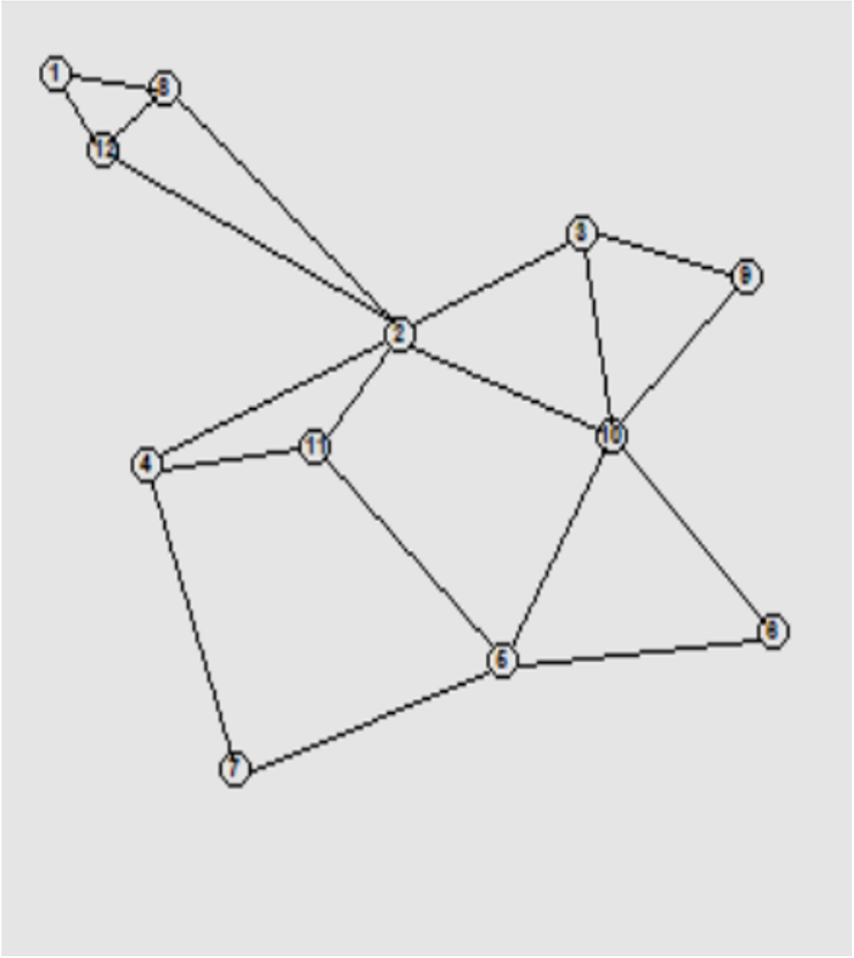


Тестування методу

Тестування проводилось за допомогою комп’ютерного моделювання у випадково згенерованому двунаправленому графові, вершина кожного є вузлом робочої мережі,

Нумерація вершин, присвоєння координат відбувалась випадково, відповідно до DSDV протоколу.
Використано генератор трафіку CBR як зовнішня бібліотека.

Мережа згенерована у вигляді колекцій типу вершин вузлів та ребер з інкапсульованою логікою обміну трафіку, та мережі графу.



Порівняння результатів дослідження

Таблиця 1.

	Величина	Мережа з використанням AODV	Мережа з модифікаціями
Затримки при передачі від джерела до одержувача	50 kb/s	0.0157 s Delay	0.0142 s Delay
	100 kb/s	0.0276 s Delay	0.0271 s Delay

Результати тестування запропонованого методу

- При стисканні заголовків передачі даних методом HPACK досягнуто зменшення службового трафіку в середньому на 11%.
- Для «гарячих» вузлів застосовано request-pipe на рівні обробки черги повідомлень, що зменшило кількість створених зациклень в мережі на 25%.
- Вузол відслідковує чергу та ігнорує дублюючі повідомлення шляхом попереднього ідентифікування кожного бінарним значенням, що складається з timestamp (4 байти), ідентификатора вузла (3 байти) та лічільника (3 байти).
- Така модель дозволяє створювати до 220 унікальних повідомлень в секунду. Після сегментування пакету та ідентифікатора можна зрівняти пакети та ідентифікатори на дублювання.

Обмеження запропонованого методу при передачі даних в умовах швидкого розгортання та пошуку резервного каналу зв'язку

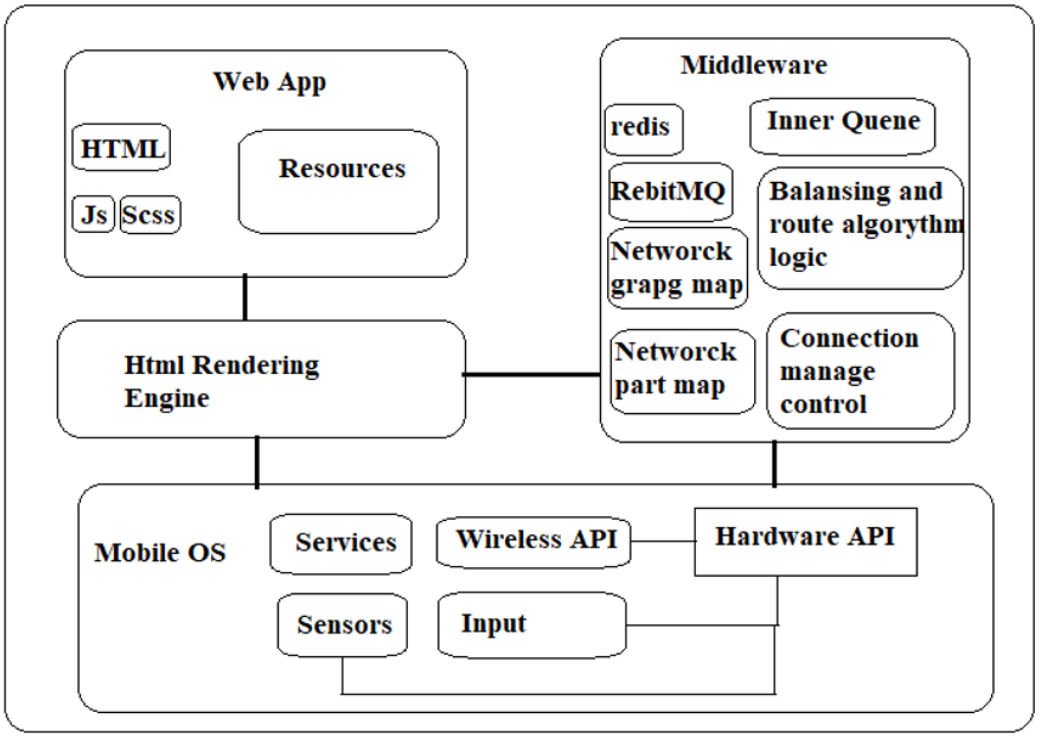
Обмеженням для роботи алгоритму було встановлення правила:

$$D_B(A) > \ldots > D_B(N) > \ldots > D_B(B); \quad (1)$$

$$S_B(A) \le \ldots \le S_B(N) \le \ldots \le S_B(B), \quad (2)$$

для кожного вузла мережі , де $D_B(N)$ – це відстань до вузла B в деякій метриці, відомій вузлу N, а $S_B(N)$ – це міра актуальності інформації про маршрут до вузла B, відомому вузлу N. З табл.1 ми бачимо, що при середніх навантаженнях приріст у швидкодії в 11-12%, при навантаженнях до 30 kb/s та після 130 kb/s – затримка у вершині більша. Тобто для ситуацій з нормальною завантаженості мережі застосування покращень є доцільним.

Особливості архітектури програмного комплексу



Використані технології

- Redis
- Android Native API
- Java
- Js
- “Fast Android Networking”
- kernel Sockets Api
- Xamarin
- Near P2P
- TinkerPop

Наукова новизна

Запропонована модифікація програмного методу безперервної передачі даних, що відрізняється від існуючих методів врахуванням зміни відношення вузлів в мережі додавання мультиплексування передачі даних між вузлами, що дозволяє збільшити точність при побудові таблиць маршрутизації при передачі даних.

БІЗНЕС-МОДЕЛЬ

<table>
<tr><th>Ключові партнери</th><th>Ключова діяльність</th><th>Ціннісна пропозиція</th><th>Відношення з клієнтами</th><th>Сегменти користувачів</th></tr>
<tr>
<td rowspan="3">1. Користувачі смартфонів
2. Телекомунікаційні компанії
3. Логістичні компанії
4. Водії</td>
<td>1. Забезпечення безперервного зв’язку в місцях з послабленим сигналом</td>
<td rowspan="3">1. Забезпечення безперервного отримання трафіку в районах з обмеженим доступом до мережі основної мережі
2. Покращення швидкодії обробки трафіку у компаній
3. Простота використання на інших платформах</td>
<td>1. Побудова лояльності</td>
<td rowspan="3">1. Дослідницькі центри
2. Логістичний бізнес
3. Телекомунікаційний бізнес
4. Фізичні особи, що мають Android смартфон та активно використовують мобільну інтернет мережу</td>
</tr>
<tr><th>Ключові ресурси</th><th>Канали</th></tr>
<tr><td>1. Розробники
2. Android пристрої
3. Спеціалізоване обладнання.</td><td>1. Соціальні мережі.
2. Контекстна реклама.</td></tr>
<tr><th colspan="3">Структура витрат</th><th colspan="2">Джерела доходу</th></tr>
<tr><td colspan="3">1. Фіксовані витрати (оренда приміщення та обладнання, зарплата персоналу)
2. Змінні витрати (придбання оборотних коштів)</td><td colspan="2">Мікротранзації, Організація трафіку, Контекстний аналіз на основі трафіку та або на основі місцеположення користувача, Програмна автоматизація алгоритмів Телекомукаційного бізнесу.</td></tr>
</table>

ФІНАНСОВИЙ ПЛАН СТАРТАПУ

ДОХОДИ (грн)	Довготривалі підписки	Міктротранзакції	Продаж ліцензійних ключів	Розширена технічна підтримка	Всього
Сумарно за рік:	200000	80000	100500	20000	400500

ВИТРАТИ (грн)	Технічна підтримка	Оплата праці	Оренда та комунальні витрати	Реклама	Всього
Сумарно за рік:	20000	155000	20000	87000	282500

Висновки

1. З’ясовано оптимальні способи безперервної передачі даних на основі бездротових каналів зв’язку в умовах швидкого розгортання та пошуку резервного каналу зв’язку, а саме використання ad hoc мереж, зокрема, MANET, з використанням AODV протоколу.
2. Проведено вибір та аналіз варіантів покращення базового методу, проведено досліження особливостей методу.
3. Розроблено та систематизовано покращення базового методу та імплементовано модифікований метод в програмне забезпечення на основі Android пристроїв.
4. У запропонованому методі збільшено швидкодію передачі даних за рахунок зменшення втрати пакетів та зменшення повторних надсилань. Здійснено моделювання бездротової мережі з підтримкою стану втрати мінімальної кількості пакетів під час обміну інформацією. Здійснено тестування та порівняння результатів дослідження. Зменшено затримку в рамках вузла на 10%.
5. Отримані результати можуть використовуватися для забезпечення безперервної передачі даних між рухомими об’єктами у режимі реального часу.
6. На основі результатів магістерської дисертації розроблено стартап проект.

Апробація

1.XI наукова конференція магістрантів та аспірантів «Прикладна математика та комп’ютинг» (ПМК-2018-2).

Дякую за увагу!

Додаток 2
Приклади програмного коду

```
package dev.edmt.chatapp;

import android.content.Context;
import android.support.test.InstrumentationRegistry;
import android.support.test.runner.AndroidJUnit4;

import org.junit.Test;
import org.junit.runner.RunWith;

import static org.junit.Assert.*;

/**
 * Instrumentation test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test
    public void useAppContext() throws Exception {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getTargetContext();

        assertEquals("dev.edmt.chatapp", appContext.getPackageName());
    }
}
package dev.edmt.chatapp;

import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.text.format.DateFormat;

import com.firebase.ui.auth.AuthUI;
import com.firebase.ui.database.FirebaseListAdapter;
import com.github.library.bubbleview.BubbleTextView;
import com.google.android.gms.tasks.OnCompleteListener;
```

```
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.FirebaseDatabase;

import hani.momanii.supernova_emoji_library.Actions.EmojIconActions;
import hani.momanii.supernova_emoji_library.Helper.EmojiconEditText;
import hani.momanii.supernova_emoji_library.Helper.EmojiconTextView;

public class MainActivity extends AppCompatActivity {

    private static int SIGN_IN_REQUEST_CODE = 1;
    private FirebaseListAdapter<ChatMessage> adapter;
    RelativeLayout activity_main;

    //Add Emojicon
    EmojiconEditText emojiconEditText;
    ImageView emojiButton,submitButton;
    EmojIconActions emojIconActions;


    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if(item.getItemId() == R.id.menu_sign_out)
        {
            AuthUI.getInstance().signOut(this).addOnCompleteListener(new
OnCompleteListener<Void>() {
                @Override
                public void onComplete(@NonNull Task<Void> task) {
                    Snackbar.make(activity_main,"You have been signed out.",
Snackbar.LENGTH_SHORT).show();
                    finish();
                }
            });
        }
        return true;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main_menu,menu);
        return true;
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode == SIGN_IN_REQUEST_CODE)
        {
            if(resultCode == RESULT_OK)
            {
                Snackbar.make(activity_main,"Successfully signed in.Welcome!",
Snackbar.LENGTH_SHORT).show();
```

```
            displayChatMessage();
        }
        else{
            Snackbar.make(activity_main,"We couldn't sign you in.Please try again later",
Snackbar.LENGTH_SHORT).show();
            finish();
        }
    }
  }

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    activity_main = (RelativeLayout)findViewById(R.id.activity_main);

    //Add Emoji
    emojiButton = (ImageView)findViewById(R.id.emoji_button);
    submitButton = (ImageView)findViewById(R.id.submit_button);
    emojiconEditText = (EmojiconEditText)findViewById(R.id.emojicon_edit_text);
    emojIconActions = new
EmojIconActions(getApplicationContext(),activity_main,emojiButton,emojiconEditText);
    emojIconActions.ShowEmojicon();

    submitButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            FirebaseDatabase.getInstance().getReference().push().setValue(new
ChatMessage(emojiconEditText.getText().toString(),
                    FirebaseAuth.getInstance().getCurrentUser().getEmail()));
            emojiconEditText.setText("");
            emojiconEditText.requestFocus();
        }
    });

    //Check if not sign-in then navigate Signin page
    if(FirebaseAuth.getInstance().getCurrentUser() == null)
    {

startActivityForResult(AuthUI.getInstance().createSignInIntentBuilder().build(),SIGN_IN_R
EQUEST_CODE);
    }
    else
    {
        Snackbar.make(activity_main,"Welcome
"+FirebaseAuth.getInstance().getCurrentUser().getEmail(),Snackbar.LENGTH_SHORT).sho
w();
        //Load content
        displayChatMessage();
    }
```

```
    }



    private void displayChatMessage() {

        ListView listOfMessage = (ListView)findViewById(R.id.list_of_message);
        adapter = new
FirebaseListAdapter<ChatMessage>(this,ChatMessage.class,R.layout.list_item,FirebaseDatab
ase.getInstance().getReference())
        {
            @Override
            protected void populateView(View v, ChatMessage model, int position) {

                //Get references to the views of list_item.xml
                TextView messageText, messageUser, messageTime;
                messageText = (BubbleTextView) v.findViewById(R.id.message_text);
                messageUser = (TextView) v.findViewById(R.id.message_user);
                messageTime = (TextView) v.findViewById(R.id.message_time);

                messageText.setText(model.getMessageText());
                messageUser.setText(model.getMessageUser());
                messageTime.setText(DateFormat.format("dd-MM-yyyy (HH:mm:ss)",
model.getMessageTime()));

            }
        };
        listOfMessage.setAdapter(adapter);
    }
}
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>

<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
</resources>
<resources>
    <string name="app_name">ChatApp</string>
</resources>
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
```

```
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

</resources>
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    >
<item android:title="Sign Out" app:showAsAction="never"
    android:id="@+id/menu_sign_out"/>
</menu>
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="dev.edmt.chatapp.MainActivity">

    <ImageView
        android:id="@+id/emoji_button"
        android:padding="4dp"
        android:src="@mipmap/smiley"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_width="40dp"
        android:layout_height="40dp" />

    <ImageView
        android:id="@+id/submit_button"
        android:padding="4dp"
        android:src="@android:drawable/ic_menu_send"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_width="40dp"
        android:layout_height="40dp" />


    <hani.momanii.supernova_emoji_library.Helper.EmojiconEditText
        android:id="@+id/emojicon_edit_text"
        android:layout_alignParentBottom="true"
        android:layout_toRightOf="@+id/emoji_button"
        android:layout_toLeftOf="@+id/submit_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:emojiconSize="28sp"
```

```
    />

    <ListView
        android:id="@+id/list_of_message"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:layout_above="@+id/emojicon_edit_text"
        android:dividerHeight="16dp"
        android:divider="@android:color/transparent"
        android:layout_marginBottom="16dp"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:stackFromBottom="true"
        android:transcriptMode="alwaysScroll"
        ></ListView>
</RelativeLayout>
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
     android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_alignParentTop="true"
        android:layout_alignParentStart="true"
        android:id="@+id/message_user"
        android:textStyle="normal|bold"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:layout_alignParentBottom="@+id/message_user"
        android:layout_alignParentEnd="true"
        android:id="@+id/message_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <com.github.library.bubbleview.BubbleTextView
        android:layout_alignParentStart="true"
        android:layout_marginTop="5dp"
        android:layout_below="@id/message_user"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1"
        android:textColor="#FFF"
        android:textSize="18sp"
        android:id="@+id/message_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="10dp"
        app:arrowWidth="8dp"
        app:angle="8dp"
        app:arrowHeight="10dp"
        app:arrowPosition="14dp"
```

```
        app:arrowLocation="left"
        app:bubbleColor="#333639"
        app:arrowCenter="true"
        />

</RelativeLayout>
```

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    buildToolsVersion '26.0.0'
    defaultConfig {
        applicationId "dev.edmt.chatapp"
        minSdkVersion 22
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:appcompat-v7:25.0.1'
    testCompile 'junit:junit:4.12'

    //Add Library
    compile 'com.android.support:design:25.0.1'
    compile 'com.firebaseui:firebase-ui:0.6.2'
    compile 'com.github.hani-momanii:SuperNova-Emoji:1.0'
    compile 'com.github.lguipeng:BubbleView:1.0.1'
}
apply plugin: 'com.google.gms.google-services'
```

```
{
  "project_info": {
    "project_number": "279433878197",
    "firebase_url": "https://xamarinchatapp-35878.firebaseio.com",
    "project_id": "xamarinchatapp-35878",
    "storage_bucket": "xamarinchatapp-35878.appspot.com"
  },
  "client": [
```

```
   {
    "client_info": {
     "mobilesdk_app_id": "1:279433878197:android:9f731b30a3f6c525",
     "android_client_info": {
      "package_name": "dev.edmt.chatapp"
     }
    },
    "oauth_client": [
     {
      "client_id": "279433878197-
qooi4sh2ghg1dbhk2339n8a7csn8pau3.apps.googleusercontent.com",
      "client_type": 1,
      "android_info": {
       "package_name": "dev.edmt.chatapp",
       "certificate_hash": "B2A5D0D2FBDA1976D9EA02862DD4BEDF6CFD50E1"
      }
     },
     {
      "client_id": "279433878197-
5tb8iahnquk075od67v0i9m659g257f4.apps.googleusercontent.com",
      "client_type": 3
     }
    ],
    "api_key": [
     {
      "current_key": "AIzaSyBVa8RYVf9ISfhCg0zuStLQj8rl1WeE6mc"
     }
    ],
    "services": {
     "analytics_service": {
      "status": 1
     },
     "appinvite_service": {
      "status": 2,
      "other_platform_oauth_client": [
       {
        "client_id": "279433878197-
5tb8iahnquk075od67v0i9m659g257f4.apps.googleusercontent.com",
        "client_type": 3
       }
      ]
     },
     "ads_service": {
      "status": 2
     }
    }
   }
  ],
  "configuration_version": "1"
}
// Top-level build file where you can add configuration options common to all sub-
projects/modules.
```

```
buildscript {
    repositories {
        jcenter()
        google()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.2.1'
        classpath 'com.google.gms:google-services:3.1.1'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        jcenter()
        maven {url "https://jitpack.io"}
        google()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
#!/usr/bin/env bash

##############################################################################
##
##  Gradle start up script for UN*X
##
##############################################################################

# Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to
pass JVM options to this script.
DEFAULT_JVM_OPTS=""

APP_NAME="Gradle"
APP_BASE_NAME=`basename "$0"`

# Use the maximum available, or set MAX_FD != -1 to use that value.
MAX_FD="maximum"

warn ( ) {
    echo "$*"
}

die ( ) {
    echo
```

```
    echo "$*"
    echo
    exit 1
}

# OS specific support (must be 'true' or 'false').
cygwin=false
msys=false
darwin=false
case "`uname`" in
  CYGWIN* )
    cygwin=true
    ;;
  Darwin* )
    darwin=true
    ;;
  MINGW* )
    msys=true
    ;;
esac

# Attempt to set APP_HOME
# Resolve links: $0 may be a link
PRG="$0"
# Need this for relative symlinks.
while [ -h "$PRG" ] ; do
    ls=`ls -ld "$PRG"`
    link=`expr "$ls" : '.*-> \(.*\)$'`
    if expr "$link" : '/.*' > /dev/null; then
        PRG="$link"
    else
        PRG=`dirname "$PRG"`"/$link"
    fi
done
SAVED="`pwd`"
cd "`dirname \"$PRG\"`/" >/dev/null
APP_HOME="`pwd -P`"
cd "$SAVED" >/dev/null

CLASSPATH=$APP_HOME/gradle/wrapper/gradle-wrapper.jar

# Determine the Java command to use to start the JVM.
if [ -n "$JAVA_HOME" ] ; then
    if [ -x "$JAVA_HOME/jre/sh/java" ] ; then
        # IBM's JDK on AIX uses strange locations for the executables
        JAVACMD="$JAVA_HOME/jre/sh/java"
    else
        JAVACMD="$JAVA_HOME/bin/java"
    fi
    if [ ! -x "$JAVACMD" ] ; then
        die "ERROR: JAVA_HOME is set to an invalid directory: $JAVA_HOME
```

```
Please set the JAVA_HOME variable in your environment to match the
location of your Java installation."
    fi
else
    JAVACMD="java"
    which java >/dev/null 2>&1 || die "ERROR: JAVA_HOME is not set and no 'java'
command could be found in your PATH.

Please set the JAVA_HOME variable in your environment to match the
location of your Java installation."
fi

# Increase the maximum file descriptors if we can.
if [ "$cygwin" = "false" -a "$darwin" = "false" ] ; then
    MAX_FD_LIMIT=`ulimit -H -n`
    if [ $? -eq 0 ] ; then
        if [ "$MAX_FD" = "maximum" -o "$MAX_FD" = "max" ] ; then
            MAX_FD="$MAX_FD_LIMIT"
        fi
        ulimit -n $MAX_FD
        if [ $? -ne 0 ] ; then
            warn "Could not set maximum file descriptor limit: $MAX_FD"
        fi
    else
        warn "Could not query maximum file descriptor limit: $MAX_FD_LIMIT"
    fi
fi

# For Darwin, add options to specify how the application appears in the dock
if $darwin; then
    GRADLE_OPTS="$GRADLE_OPTS \"-Xdock:name=$APP_NAME\" \"-
Xdock:icon=$APP_HOME/media/gradle.icns\""
fi

# For Cygwin, switch paths to Windows format before running java
if $cygwin ; then
    APP_HOME=`cygpath --path --mixed "$APP_HOME"`
    CLASSPATH=`cygpath --path --mixed "$CLASSPATH"`
    JAVACMD=`cygpath --unix "$JAVACMD"`

    # We build the pattern for arguments to be converted via cygpath
    ROOTDIRSRAW=`find -L / -maxdepth 1 -mindepth 1 -type d 2>/dev/null`
    SEP=""
    for dir in $ROOTDIRSRAW ; do
        ROOTDIRS="$ROOTDIRS$SEP$dir"
        SEP="|"
    done
    OURCYGPATTERN="(^($ROOTDIRS))"
    # Add a user-defined pattern to the cygpath arguments
    if [ "$GRADLE_CYGPATTERN" != "" ] ; then
        OURCYGPATTERN="$OURCYGPATTERN|($GRADLE_CYGPATTERN)"
    fi
```

```
    # Now convert the arguments – kludge to limit ourselves to /bin/sh
    i=0
    for arg in "$@" ; do
        CHECK=`echo "$arg"|egrep -c "$OURCYGPATTERN" -`
        CHECK2=`echo "$arg"|egrep -c "^-"`                        ### Determine if an option

        if [ $CHECK -ne 0 ] && [ $CHECK2 -eq 0 ] ; then                    ### Added a condition
            eval `echo args$i`=`cygpath --path --ignore --mixed "$arg"`
        else
            eval `echo args$i`="\"$arg\""
        fi
        i=$((i+1))
    done
    case $i in
        (0) set -- ;;
        (1) set -- "$args0" ;;
        (2) set -- "$args0" "$args1" ;;
        (3) set -- "$args0" "$args1" "$args2" ;;
        (4) set -- "$args0" "$args1" "$args2" "$args3" ;;
        (5) set -- "$args0" "$args1" "$args2" "$args3" "$args4" ;;
        (6) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" ;;
        (7) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6" ;;
        (8) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6" "$args7" ;;
        (9) set -- "$args0" "$args1" "$args2" "$args3" "$args4" "$args5" "$args6" "$args7"
"$args8" ;;
    esac
fi

# Split up the JVM_OPTS And GRADLE_OPTS values into an array, following the shell
quoting and substitution rules
function splitJvmOpts() {
    JVM_OPTS=("$@")
}
eval splitJvmOpts $DEFAULT_JVM_OPTS $JAVA_OPTS $GRADLE_OPTS
JVM_OPTS[${#JVM_OPTS[*]}]="-Dorg.gradle.appname=$APP_BASE_NAME"

exec "$JAVACMD" "${JVM_OPTS[@]}" -classpath "$CLASSPATH"
org.gradle.wrapper.GradleWrapperMain "$@"

package com.layer.messenger;

import android.os.Bundle;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuItem;

import com.layer.sdk.LayerClient;
import com.squareup.picasso.Picasso;

public abstract class BaseActivity extends AppCompatActivity {
    private final int mLayoutResId;
```

```
    private final int mMenuResId;
    private final int mMenuTitleResId;
    private final boolean mMenuBackEnabled;

    public BaseActivity(int layoutResId, int menuResId, int menuTitleResId, boolean
menuBackEnabled) {
        mLayoutResId = layoutResId;
        mMenuResId = menuResId;
        mMenuTitleResId = menuTitleResId;
        mMenuBackEnabled = menuBackEnabled;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(mLayoutResId);

        ActionBar actionBar = getSupportActionBar();
        if (actionBar == null) return;
        if (mMenuBackEnabled) actionBar.setDisplayHomeAsUpEnabled(true);
        actionBar.setTitle(mMenuTitleResId);
    }

    @Override
    public void setTitle(CharSequence title) {
        ActionBar actionBar = getSupportActionBar();
        if (actionBar == null) {
            super.setTitle(title);
        } else {
            actionBar.setTitle(title);
        }
    }

    @Override
    public void setTitle(int titleId) {
        ActionBar actionBar = getSupportActionBar();
        if (actionBar == null) {
            super.setTitle(titleId);
        } else {
            actionBar.setTitle(titleId);
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        LayerClient client = App.getLayerClient();
        if (client == null) return;
        if (client.isAuthenticated()) {
            client.connect();
        } else {
            client.authenticate();
```

```
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(mMenuResId, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == android.R.id.home) {
            // Menu "Navigate Up" acts like hardware back button
            onBackPressed();
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

    protected LayerClient getLayerClient() {
        return App.getLayerClient();
    }

    protected Picasso getPicasso() {
        return App.getPicasso();
    }
}
package com.layer.messenger;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.inputmethod.EditorInfo;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;

import com.layer.atlas.AtlasAvatar;
import com.layer.atlas.util.IdentityDisplayNameComparator;
```

```
import com.layer.messenger.util.Util;
import com.layer.sdk.LayerClient;
import com.layer.sdk.changes.LayerChangeEvent;
import com.layer.sdk.listeners.LayerChangeEventListener;
import com.layer.sdk.listeners.LayerPolicyListener;
import com.layer.sdk.messaging.Conversation;
import com.layer.sdk.messaging.Identity;
import com.layer.sdk.policy.Policy;

import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class ConversationSettingsActivity extends BaseActivity implements
LayerPolicyListener, LayerChangeEventListener {
    private EditText mConversationName;
    private Switch mShowNotifications;
    private RecyclerView mParticipantRecyclerView;
    private Button mLeaveButton;
    private Button mAddParticipantsButton;

    private Conversation mConversation;
    private ParticipantAdapter mParticipantAdapter;

    public ConversationSettingsActivity() {
        super(R.layout.activity_conversation_settings, R.menu.menu_conversation_details,
R.string.title_conversation_details, true);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mConversationName = (EditText) findViewById(R.id.conversation_name);
        mShowNotifications = (Switch) findViewById(R.id.show_notifications_switch);
        mParticipantRecyclerView = (RecyclerView) findViewById(R.id.participants);
        mLeaveButton = (Button) findViewById(R.id.leave_button);
        mAddParticipantsButton = (Button) findViewById(R.id.add_participant_button);

        // Get Conversation from Intent extras
        Uri conversationId =
getIntent().getParcelableExtra(PushNotificationReceiver.LAYER_CONVERSATION_KEY);
        mConversation = getLayerClient().getConversation(conversationId);
        if (mConversation == null && !isFinishing()) finish();

        initializeMarkAsReadButton();

        mParticipantAdapter = new ParticipantAdapter();
        mParticipantRecyclerView.setAdapter(mParticipantAdapter);
```

```
        LinearLayoutManager manager = new LinearLayoutManager(this,
LinearLayoutManager.VERTICAL, false);
        mParticipantRecyclerView.setLayoutManager(manager);

        mConversationName.setOnEditorActionListener(new
TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
                if (actionId == EditorInfo.IME_ACTION_DONE || (event != null &&
event.getAction() == KeyEvent.ACTION_DOWN && event.getKeyCode() ==
KeyEvent.KEYCODE_ENTER)) {
                    String title = ((EditText) v).getText().toString().trim();
                    Util.setConversationMetadataTitle(mConversation, title);
                    Toast.makeText(v.getContext(), R.string.toast_group_name_updated,
Toast.LENGTH_SHORT).show();
                    return true;
                }
                return false;
            }
        });

        mShowNotifications.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                PushNotificationReceiver.getNotifications(ConversationSettingsActivity.this)
                        .setEnabled(mConversation.getId(), isChecked);
            }
        });

        mLeaveButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                setEnabled(false);
                mConversation.removeParticipants(getLayerClient().getAuthenticatedUser());
                refresh();
                Intent intent = new Intent(ConversationSettingsActivity.this,
ConversationsListActivity.class);
                intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                setEnabled(true);
                ConversationSettingsActivity.this.startActivity(intent);
            }
        });

        mAddParticipantsButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // TODO
                Toast.makeText(v.getContext(), "Coming soon", Toast.LENGTH_LONG).show();
            }
        });
    }
```

```
    public void setEnabled(boolean enabled) {
        mShowNotifications.setEnabled(enabled);
        mLeaveButton.setEnabled(enabled);
    }

    private void refresh() {
        if (!getLayerClient().isAuthenticated()) return;

        mConversationName.setText(Util.getConversationMetadataTitle(mConversation));

mShowNotifications.setChecked(PushNotificationReceiver.getNotifications(this).isEnabled(
mConversation.getId()));

        Set<Identity> participantsMinusMe = new HashSet<>(mConversation.getParticipants());
        participantsMinusMe.remove(getLayerClient().getAuthenticatedUser());

        if (participantsMinusMe.size() == 0) {
            // I've been removed
            mConversationName.setEnabled(false);
            mLeaveButton.setVisibility(View.GONE);
        } else if (participantsMinusMe.size() == 1) {
            // 1-on-1
            mConversationName.setEnabled(false);
            mLeaveButton.setVisibility(View.GONE);
        } else {
            // Group
            mConversationName.setEnabled(true);
            mLeaveButton.setVisibility(View.VISIBLE);
        }
        mParticipantAdapter.refresh();
    }

    @Override
    protected void onResume() {
        super.onResume();
        getLayerClient().registerPolicyListener(this).registerEventListener(this);
        setEnabled(true);
        refresh();
    }

    @Override
    protected void onPause() {
        getLayerClient().unregisterPolicyListener(this).unregisterEventListener(this);
        super.onPause();
    }

    @Override
    public void onPolicyListUpdate(LayerClient layerClient, List<Policy> list, List<Policy>
list1) {
        refresh();
    }
```

```
    @Override
    public void onChangeEvent(LayerChangeEvent layerChangeEvent) {
        refresh();
    }

    private void initializeMarkAsReadButton() {
        Button markAllReadButton = (Button) findViewById(R.id.mark_all_read_button);
        if (mConversation.isReadReceiptsEnabled()) {
            markAllReadButton.setVisibility(View.VISIBLE);
            markAllReadButton.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    mConversation.markAllMessagesAsRead();
                }
            });
        } else {
            markAllReadButton.setVisibility(View.GONE);
        }
    }

    private class ParticipantAdapter extends RecyclerView.Adapter<ViewHolder> {
        List<Identity> mParticipants = new ArrayList<>();

        public void refresh() {
            // Get new sorted list of Participants
            mParticipants.clear();
            Set<Identity> conversationParticipants = mConversation.getParticipants();
            conversationParticipants.remove(getLayerClient().getAuthenticatedUser());
            mParticipants.addAll(conversationParticipants);
            Collections.sort(mParticipants, new IdentityDisplayNameComparator());

            // Adjust participant container height
            int height = Math.round(mParticipants.size() *
getResources().getDimensionPixelSize(com.layer.atlas.R.dimen.atlas_secondary_item_height
));
            LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, height);
            mParticipantRecyclerView.setLayoutParams(params);

            // Notify changes
            notifyDataSetChanged();
        }

        @Override
        public ViewHolder onCreateViewHolder(final ViewGroup parent, int viewType) {
            ViewHolder viewHolder = new ViewHolder(parent);
            viewHolder.mAvatar.init(App.getPicasso());
            viewHolder.itemView.setTag(viewHolder);

            // Click to display remove / block dialog
            viewHolder.itemView.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                final ViewHolder holder = (ViewHolder) v.getTag();

                AlertDialog.Builder builder = new AlertDialog.Builder(v.getContext())
                        .setMessage(holder.mTitle.getText().toString());

                if (mConversation.getParticipants().size() > 2) {
                    builder.setNeutralButton(R.string.alert_button_remove, new
DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            mConversation.removeParticipants(holder.mParticipant);
                        }
                    });
                }

                builder.setPositiveButton(holder.mBlockPolicy != null ?
R.string.alert_button_unblock : R.string.alert_button_block,
                        new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                Identity participant = holder.mParticipant;
                                if (holder.mBlockPolicy == null) {
                                    // Block
                                    holder.mBlockPolicy = new
Policy.Builder(Policy.PolicyType.BLOCK).sentByUserId(participant.getUserId()).build();
                                    getLayerClient().addPolicy(holder.mBlockPolicy);
                                    holder.mBlocked.setVisibility(View.VISIBLE);
                                } else {
                                    getLayerClient().removePolicy(holder.mBlockPolicy);
                                    holder.mBlockPolicy = null;
                                    holder.mBlocked.setVisibility(View.INVISIBLE);
                                }
                            }
                        }).setNegativeButton(R.string.alert_button_cancel, new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which) {
                        dialog.dismiss();
                    }
                }).show();
            }
        });

        return viewHolder;
    }

    @Override
    public void onBindViewHolder(ViewHolder viewHolder, int position) {
        Identity participant = mParticipants.get(position);
        viewHolder.mTitle.setText(com.layer.atlas.util.Util.getDisplayName(participant));
```

```
            viewHolder.mAvatar.setParticipants(participant);
            viewHolder.mParticipant = participant;

            Policy block = null;
            for (Policy policy : getLayerClient().getPolicies()) {
                if (policy.getPolicyType() != Policy.PolicyType.BLOCK) continue;
                if (!policy.getSentByUserID().equals(participant.getUserId())) continue;
                block = policy;
                break;
            }

            viewHolder.mBlockPolicy = block;
            viewHolder.mBlocked.setVisibility(block == null ? View.INVISIBLE :
View.VISIBLE);
        }

        @Override
        public int getItemCount() {
            return mParticipants.size();
        }
    }

    private static class ViewHolder extends RecyclerView.ViewHolder {
        AtlasAvatar mAvatar;
        TextView mTitle;
        ImageView mBlocked;
        Identity mParticipant;
        Policy mBlockPolicy;

        public ViewHolder(ViewGroup parent) {
            super(LayoutInflater.from(parent.getContext()).inflate(R.layout.participant_item,
parent, false));
            mAvatar = (AtlasAvatar) itemView.findViewById(R.id.avatar);
            mTitle = (TextView) itemView.findViewById(R.id.title);
            mBlocked = (ImageView) itemView.findViewById(R.id.blocked);
        }
    }
}
public class LoginActivity extends AppCompatActivity {
    EditText mEmail;
    EditText mPassword;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login_rails);
        final ActionBar actionBar = getSupportActionBar();
        if (actionBar != null) actionBar.hide();

        mEmail = (EditText) findViewById(R.id.email);
        mEmail.setImeOptions(EditorInfo.IME_ACTION_NEXT);
```

```
        mPassword = (EditText) findViewById(R.id.password);
        mPassword.setImeOptions(EditorInfo.IME_ACTION_DONE);
        mPassword.setOnEditorActionListener(new TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
                if (actionId == EditorInfo.IME_ACTION_DONE || (event != null &&
event.getKeyCode() == KeyEvent.KEYCODE_ENTER)) {
                    final String email = mEmail.getText().toString().trim();
                    if (email.isEmpty()) return true;

                    final String password = mPassword.getText().toString().trim();
                    if (password.isEmpty()) return true;

                    login(email, password);
                    return true;
                }
                return false;
            }
        });

        // Optionally add a CustomEndpoint Spinner (not typical)
        if (App.LAYER_APP_ID == null) {
            Spinner customEndpoints = CustomEndpoint.createSpinner(this);
            if (customEndpoints != null) {
                customEndpoints.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
                ((ViewGroup) mPassword.getParent()).addView(customEndpoints);
            }
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
        mEmail.setEnabled(true);
        mPassword.setEnabled(true);
    }

    private void login(final String email, final String password) {
        if (Log.isPerfLoggable()) {
            Log.perf("LoginActivity performing login attempt");
        }

        mEmail.setEnabled(false);
        mPassword.setEnabled(false);
        final ProgressDialog progressDialog = new ProgressDialog(LoginActivity.this);
        progressDialog.setMessage(getResources().getString(R.string.login_dialog_message));
        progressDialog.show();
        App.authenticate(new LayerAuthenticationProvider.Credentials(App.getLayerAppId(),
email, password, null),
                new AuthenticationProvider.Callback() {
```

```
                @Override
                public void onSuccess(AuthenticationProvider provider, String userId) {
                    if (Log.isPerfLoggable()) {
                        Log.perf("LoginActivity received success callback for login attempt");
                    }

                    provider.setCallback(null);
                    progressDialog.dismiss();
                    if (Log.isLoggable(Log.VERBOSE)) {
                        Log.v("Successfully authenticated as `" + email + "` with userId `" + userId
+ "`");
                    }
                    Intent intent = new Intent(LoginActivity.this, ConversationsListActivity.class);
                    intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_NEW_TASK);
                    LoginActivity.this.startActivity(intent);
                }

                @Override
                public void onError(AuthenticationProvider provider, final String error) {
                    progressDialog.dismiss();
                    if (Log.isLoggable(Log.ERROR)) {
                        Log.e("Failed to authenticate as `" + email + "`: " + error);
                    }

                    provider.setCallback(null);
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            Toast.makeText(LoginActivity.this, error,
Toast.LENGTH_LONG).show();
                            mEmail.setEnabled(true);
                            mPassword.setEnabled(true);
                        }
                    });
                }
            });
    }
}

import java.util.concurrent.atomic.AtomicInteger;

public class PushNotificationReceiver extends BroadcastReceiver {
    public final static int MESSAGE_ID = 1;
    private final static AtomicInteger sPendingIntentCounter = new AtomicInteger(0);
    private static Notifications sNotifications;

    public final static String ACTION_PUSH = "com.layer.sdk.PUSH";
    public final static String ACTION_CANCEL = BuildConfig.APPLICATION_ID +
".CANCEL_PUSH";

    public final static String LAYER_CONVERSATION_KEY = "layer-conversation-id";
```

```
    public final static String LAYER_MESSAGE_KEY = "layer-message-id";

    /**
     * Parses the `com.layer.sdk.PUSH` Intent.
     */
    @Override
    public void onReceive(final Context context, Intent intent) {
        Bundle extras = intent.getExtras();
        if (extras == null) return;

        final PushNotificationPayload payload =
PushNotificationPayload.fromLayerPushExtras(extras);
        final Uri conversationId = extras.getParcelable(LAYER_CONVERSATION_KEY);
        final Uri messageId = extras.getParcelable(LAYER_MESSAGE_KEY);

        if (intent.getAction().equals(ACTION_PUSH)) {
            // New push from Layer
            if (Log.isLoggable(Log.VERBOSE)) Log.v("Received notification for: " +
messageId);
            if (messageId == null) {
                if (Log.isLoggable(Log.ERROR)) Log.e("No message to notify: " + extras);
                return;
            }
            if (conversationId == null) {
                if (Log.isLoggable(Log.ERROR)) Log.e("No conversation to notify: " + extras);
                return;
            }

            if (!getNotifications(context).isEnabled()) {
                if (Log.isLoggable(Log.VERBOSE)) {
                    Log.v("Blocking notification due to global app setting");
                }
                return;
            }

            if (!getNotifications(context).isEnabled(conversationId)) {
                if (Log.isLoggable(Log.VERBOSE)) {
                    Log.v("Blocking notification due to conversation detail setting");
                }
                return;
            }

            // Try to have content ready for viewing before posting a Notification
            LayerClient layerClient = App.getLayerClient();

            if (layerClient != null) {
                layerClient.waitForContent(messageId, new
LayerClient.ContentAvailableCallback() {
                    @Override
                    public void onContentAvailable(LayerClient client, @NonNull Queryable object)
{
                        if (Log.isLoggable(Log.VERBOSE)) {
```

```
                    Log.v("Pre-fetched notification content");
                }
                getNotifications(context).add(context, (Message) object, payload.getText());
            }

            @Override
            public void onContentFailed(LayerClient client, Uri objectId, Exception e) {
                if (Log.isLoggable(Log.ERROR)) {
                    Log.e("Failed to fetch notification content");
                }
                getNotifications(context).notifyOnContentFailure(context, conversationId,
messageId, payload.getText());
            }
        });
    }

} else if (intent.getAction().equals(ACTION_CANCEL)) {
    // User swiped notification out
    if (Log.isLoggable(Log.VERBOSE)) {
        Log.v("Cancelling notifications for: " + conversationId);
    }
    getNotifications(context).clear(conversationId);
} else {
    if (Log.isLoggable(Log.ERROR)) {
        Log.e("Got unknown intent action: " + intent.getAction());
    }
}
}

public static synchronized Notifications getNotifications(Context context) {
    if (sNotifications == null) {
        sNotifications = new Notifications(context);
    }
    return sNotifications;
}

/**
 * Notifications manages notifications displayed on the user's device.  Notifications are
 * grouped by Conversation, where a Conversation's notifications are rolled-up into single
 * notification summaries.
 */
public static class Notifications {
    private static final String KEY_ALL = "all";
    private static final String KEY_POSITION = "position";
    private static final String KEY_TEXT = "text";

    private final int MAX_MESSAGES = 5;
    // Contains black-listed conversation IDs and the global "all" key for notifications
    private final SharedPreferences mDisableds;

    // Contains positions for message IDs
    private final SharedPreferences mPositions;
```

```
    private final SharedPreferences mMessages;
    private final NotificationManager mManager;

    public Notifications(Context context) {
        mDisableds = context.getSharedPreferences("notification_disableds",
Context.MODE_PRIVATE);
        mPositions = context.getSharedPreferences("notification_positions",
Context.MODE_PRIVATE);
        mMessages = context.getSharedPreferences("notification_messages",
Context.MODE_PRIVATE);
        mManager = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
    }

    public boolean isEnabled() {
        return !mDisableds.contains(KEY_ALL);
    }

    public boolean isEnabled(Uri conversationId) {
        if (conversationId == null) {
            return isEnabled();
        }
        return !mDisableds.contains(conversationId.toString());
    }

    public void setEnabled(boolean enabled) {
        if (enabled) {
            mDisableds.edit().remove(KEY_ALL).apply();
        } else {
            mDisableds.edit().putBoolean(KEY_ALL, true).apply();
            mManager.cancelAll();
        }
    }

    public void setEnabled(Uri conversationId, boolean enabled) {
        if (conversationId == null) {
            return;
        }
        if (enabled) {
            mDisableds.edit().remove(conversationId.toString()).apply();
        } else {
            mDisableds.edit().putBoolean(conversationId.toString(), true).apply();
            mManager.cancel(conversationId.toString(), MESSAGE_ID);
        }
    }

    public void clear(final Conversation conversation) {
        if (conversation == null) return;
        clear(conversation.getId());
    }

    /**
```

```
     * Called when a Conversation is opened or message is marked as read
     * Clears messages map; sets position to greatest position
     *
     * @param conversationId Conversation whose notifications should be cleared
     */
    public void clear(final Uri conversationId) {
        new Thread(new Runnable() {
            @Override
            public void run() {
                if (conversationId == null) return;
                String key = conversationId.toString();
                long maxPosition = getMaxPosition(conversationId);
                mMessages.edit().remove(key).commit();
                mPositions.edit().putLong(key, maxPosition).commit();
                mManager.cancel(key, MESSAGE_ID);
            }
        }).start();
    }

    /**
     * Called when a new message arrives
     *
     * @param message Message to add
     * @param text    Notification text for added Message
     */
    protected void add(Context context, Message message, String text) {
        Conversation conversation = message.getConversation();
        String key = conversation.getId().toString();
        long currentPosition = mPositions.getLong(key, Long.MIN_VALUE);

        // Ignore older messages
        if (message.getPosition() <= currentPosition) return;

        String currentMessages = mMessages.getString(key, null);

        try {
            JSONObject messages = currentMessages == null ? new JSONObject() : new
JSONObject(currentMessages);
            String messageKey = message.getId().toString();

            // Ignore if we already have this message
            if (messages.has(messageKey)) return;

            JSONObject messageEntry = new JSONObject();
            messageEntry.put(KEY_POSITION, message.getPosition());
            messageEntry.put(KEY_TEXT, text);
            messages.put(messageKey, messageEntry);

            mMessages.edit().putString(key, messages.toString()).commit();
        } catch (JSONException e) {
            if (Log.isLoggable(Log.ERROR)) {
                Log.e(e.getMessage(), e);
```

```
            }
            return;
        }
        update(context, conversation, message);
    }

    private void update(Context context, Conversation conversation, Message message) {
        String messagesString = mMessages.getString(conversation.getId().toString(), null);
        if (messagesString == null) return;

        // Get current notification texts
        Map<Long, String> positionText = new HashMap<Long, String>();
        try {
            JSONObject messagesJson = new JSONObject(messagesString);
            Iterator<String> iterator = messagesJson.keys();
            while (iterator.hasNext()) {
                String messageId = iterator.next();
                JSONObject messageJson = messagesJson.getJSONObject(messageId);
                long position = messageJson.getLong(KEY_POSITION);
                String text = messageJson.getString(KEY_TEXT);
                positionText.put(position, text);
            }
        } catch (JSONException e) {
            if (Log.isLoggable(Log.ERROR)) {
                Log.e(e.getMessage(), e);
            }
            return;
        }

        // Sort by message position
        List<Long> positions = new ArrayList<Long>(positionText.keySet());
        Collections.sort(positions);

        // Construct notification
        String conversationTitle = Util.getConversationTitle(App.getLayerClient(),
conversation);
        NotificationCompat.InboxStyle inboxStyle = new
NotificationCompat.InboxStyle().setBigContentTitle(conversationTitle);
        int i;
        if (positions.size() <= MAX_MESSAGES) {
            i = 0;
            inboxStyle.setSummaryText(null);
        } else {
            i = positions.size() - MAX_MESSAGES;
            inboxStyle.setSummaryText(context.getString(R.string.notifications_num_more,
i));
        }
        while (i < positions.size()) {
            inboxStyle.addLine(positionText.get(positions.get(i++)));
        }

        String collapsedSummary = positions.size() == 1 ? positionText.get(positions.get(0)) :
```

```
                context.getString(R.string.notifications_new_messages, positions.size());

        // Construct notification
        // TODO: use large icon based on avatars
        NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(context)
                .setSmallIcon(R.drawable.notification)
                .setContentTitle(conversationTitle)
                .setContentText(collapsedSummary)
                .setAutoCancel(true)
                .setLights(context.getResources().getColor(R.color.atlas_action_bar_background
), 100, 1900)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setDefaults(NotificationCompat.DEFAULT_SOUND |
NotificationCompat.DEFAULT_VIBRATE)
                .setStyle(inboxStyle);

        // Intent to launch when clicked
        PendingIntent clickPendingIntent = createNotificationClickIntent(context,
conversation.getId(), message.getId());
        mBuilder.setContentIntent(clickPendingIntent);

        // Intent to launch when swiped out
        PendingIntent deleteIntent = createNotificationDeleteIntent(context,
conversation.getId(), message.getId());
        mBuilder.setDeleteIntent(deleteIntent);

        // Show the notification
        mManager.notify(conversation.getId().toString(), MESSAGE_ID, mBuilder.build());
    }

    private void notifyOnContentFailure(Context context, Uri conversationId, Uri
messageId, String text) {
        NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(context)
                .setSmallIcon(R.drawable.notification)
                .setContentTitle(context.getString(R.string.push_notification_no_content_title))
                .setContentText(text)
                .setAutoCancel(true)
                .setLights(context.getResources().getColor(R.color.atlas_action_bar_background
), 100, 1900)
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setDefaults(NotificationCompat.DEFAULT_SOUND |
NotificationCompat.DEFAULT_VIBRATE);

        // Intent to launch when clicked
        PendingIntent clickPendingIntent = createNotificationClickIntent(context,
conversationId, messageId);
        mBuilder.setContentIntent(clickPendingIntent);

        // Intent to launch when swiped out
        PendingIntent deleteIntent = createNotificationDeleteIntent(context, conversationId,
messageId);
        mBuilder.setDeleteIntent(deleteIntent);
```

```
        // Show the notification
        mManager.notify(conversationId.toString(), MESSAGE_ID, mBuilder.build());
    }

    /**
     * Returns the current maximum Message position within the given Conversation, or
     * Long.MIN_VALUE if no messages are found.
     *
     * @param conversationId Conversation whose maximum Message position to return.
     * @return the current maximum Message position or Long.MIN_VALUE.
     */
    private long getMaxPosition(Uri conversationId) {
        LayerClient layerClient = App.getLayerClient();

        Query<Message> query = Query.builder(Message.class)
                .predicate(new Predicate(Message.Property.CONVERSATION,
Predicate.Operator.EQUAL_TO, conversationId))
                .sortDescriptor(new SortDescriptor(Message.Property.POSITION,
SortDescriptor.Order.DESCENDING))
                .limit(1)
                .build();

        List results = layerClient.executeQueryForObjects(query);
        if (results.isEmpty()) return Long.MIN_VALUE;
        return ((Message) results.get(0)).getPosition();
    }

    private static PendingIntent createNotificationClickIntent(Context context, Uri
conversationId, Uri messageId) {
        Intent clickIntent = new Intent(context, MessagesListActivity.class)
                .setPackage(context.getApplicationContext().getPackageName())
                .putExtra(LAYER_CONVERSATION_KEY, conversationId)
                .putExtra(LAYER_MESSAGE_KEY, messageId)
                .setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        return PendingIntent.getActivity(
                context, sPendingIntentCounter.getAndIncrement(),
                clickIntent, PendingIntent.FLAG_ONE_SHOT);
    }

    private static PendingIntent createNotificationDeleteIntent(Context context, Uri
conversationId, Uri messageId) {
        Intent cancelIntent = new Intent(ACTION_CANCEL)
                .setPackage(context.getApplicationContext().getPackageName())
                .putExtra(LAYER_CONVERSATION_KEY, conversationId)
                .putExtra(LAYER_MESSAGE_KEY, messageId);
        return PendingIntent.getBroadcast(
                context, sPendingIntentCounter.getAndIncrement(),
                cancelIntent, PendingIntent.FLAG_ONE_SHOT);
    }
  }
}
```

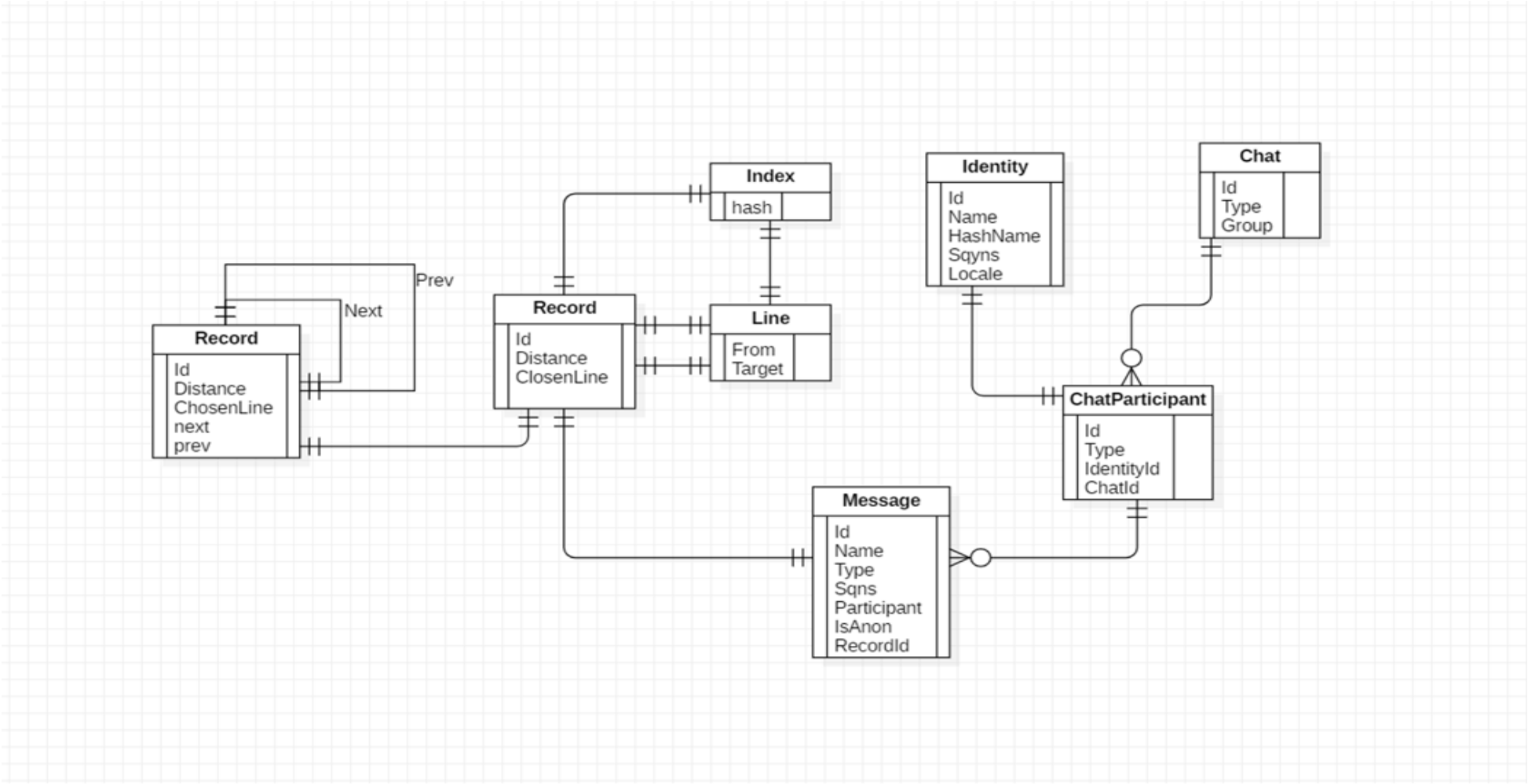
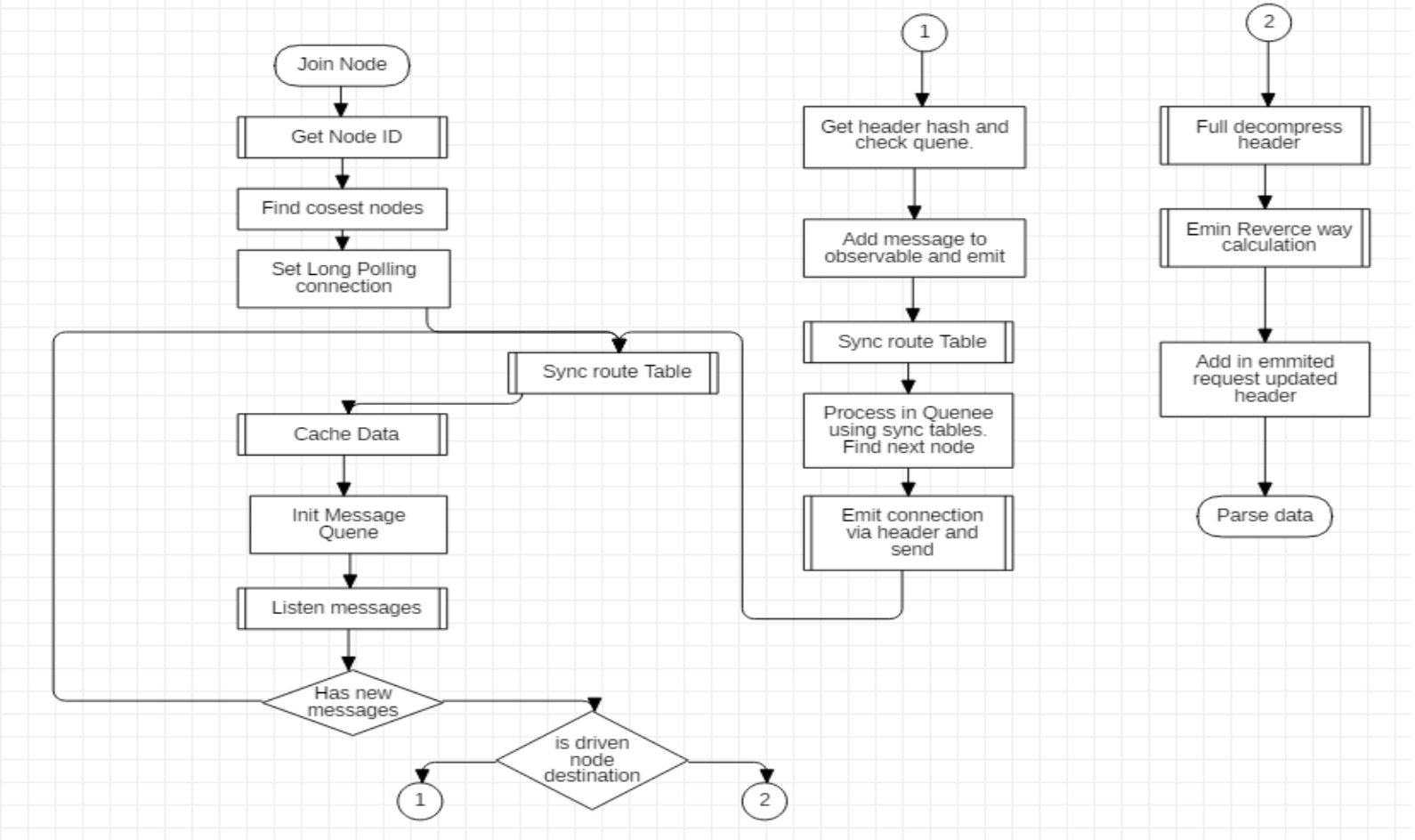


Схема бази даних
збереження потоку повідомлень
наоснові модифікованого методу
Петрусь Владислав Ігорович, КП-71мп



Модифікований алгоритм
програмної маршрутизації графу мережі
Петрусь Владислав Ігорович, КП-71мп